

msdn

magazine

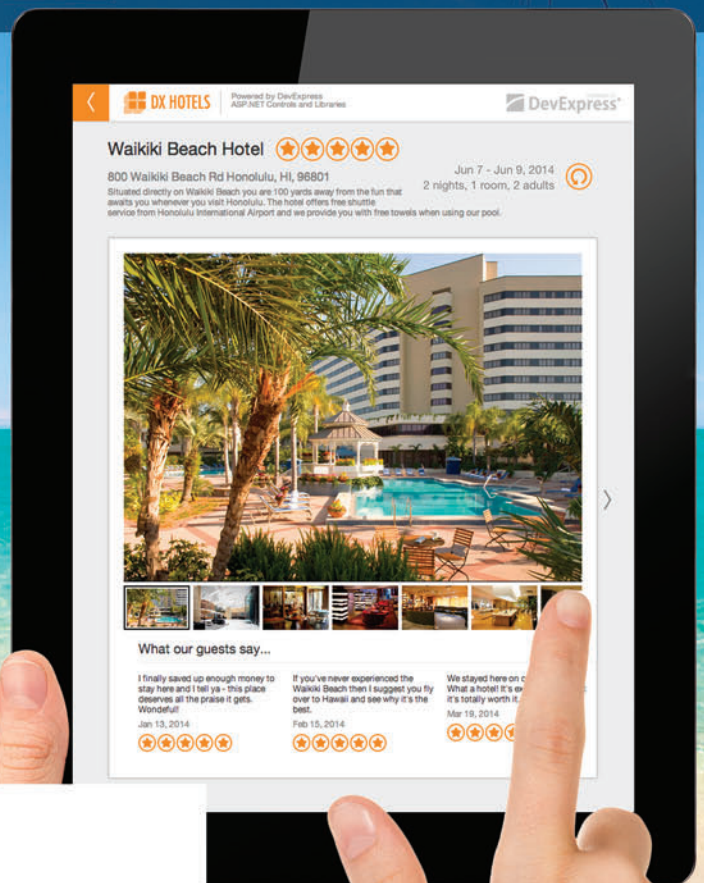


Integrate OneDrive into
Windows Store Apps.....30

Your Next Great ASP.NET App Starts Here

Deliver interactive touch-enabled user experiences for WebForms and MVC with elegant, high-performance UI controls and extensions from DevExpress.

Download your free 30-day trial at: DevExpress.com/ASP

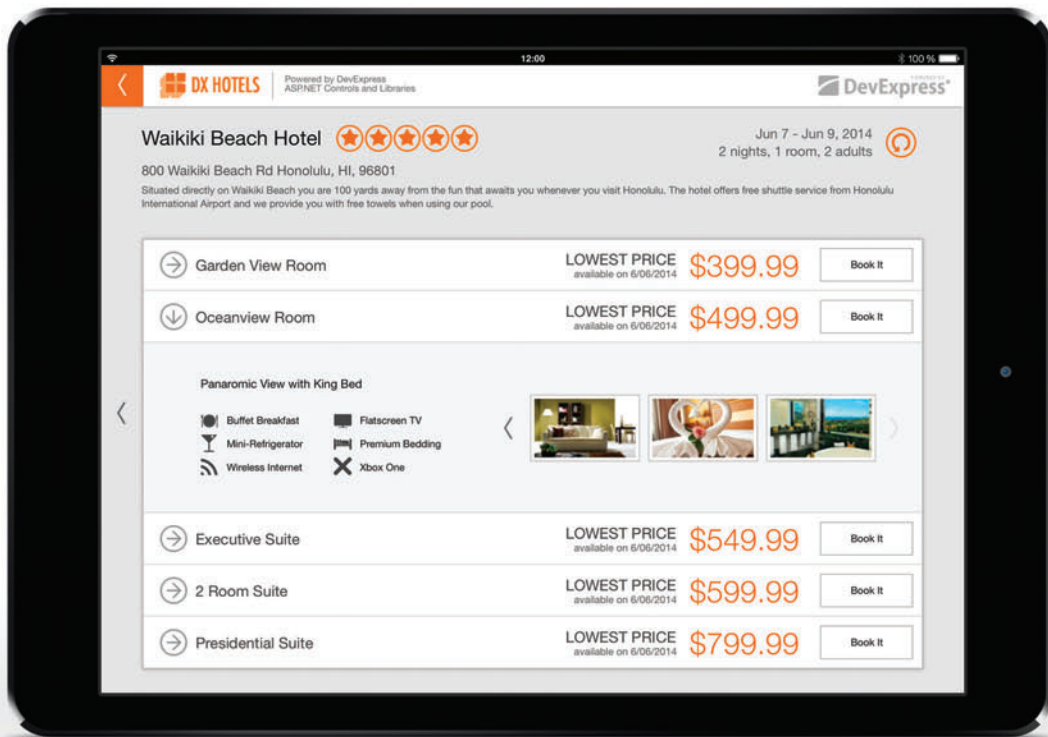


 **DevExpress®**



Become a UI Superhero

Learn More at DevExpress.com/Superhero



msdn

magazine



Integrate OneDrive into
Windows Store Apps.....30

Integrating OneDrive into Your Windows Store Apps Tony Champion	30
Developing Your First Game with Unity and C# Adam Tuliper	36
Build MVVM Apps with Xamarin and MvvmCross Thomas LeBrun	48
Microsoft Azure Media Services Gregory Prentice	54
Use Distributed Cache in Microsoft Azure Iqbal M. Khan and Jeremiah Talkar	64

COLUMNS

CUTTING EDGE

Documents, Databases
and Eventual Consistency
Dino Esposito, page 8

WINDOWS WITH C++

DirectComposition:
A Retained-Mode API
to Rule Them All
Kenny Kerr, page 12

DATA POINTS

Cool (and Free) Tools
for Entity Framework
Julie Lerman, page 24

TEST RUN

Solving Sudoku Puzzles
Using the MSF Library
James McCaffrey, page 74

DON'T GET ME STARTED

Unwritten Rules
David Platt, page 80



Microsoft

#1

**1: As a Scrum team, I want to
get more \$#*% done**

Assigned To: Your Team

Priority: Very High

Release: V 1.0

0 sp  25 sp

INCREASE YOUR TEAM'S VELOCITY AND GET MORE DONE WITH THE #1 SCRUM TOOL.

You can do better than sticky notes. **Axosoft Scrum** is the easiest way to manage backlogs, plan releases and analyze burndown charts. Learn more at [Axosoft.com/MSDNscrum](https://axosoft.com/MSDNscrum).



\$1



A BUG TRACKER FOR YOUR WHOLE TEAM NOW COSTS THE SAME AS A DOLLAR MENU BURGER.

Axosoft Bug Tracker is now just \$1 per year for your entire team. Not \$1 per user, but \$1 for everyone. Check it out at [Axosoft.com/MSDNbugs](https://axosoft.com/MSDNbugs).





dtSearch®

Instantly Search Terabytes of Text

25+ fielded and full-text search types

dtSearch's **own document filters** support "Office," PDF, HTML, XML, ZIP, emails (with nested attachments), and many other file types

Supports databases as well as static and dynamic websites

Highlights hits in all of the above APIs for .NET, Java, C++, SQL, etc.
64-bit and 32-bit; Win and Linux

"lightning fast" Redmond Magazine

"covers all data sources" eWeek

"results in less than a second" InfoWorld

hundreds more reviews and developer case studies at www.dtsearch.com

dtSearch products:

Desktop with Spider	Web with Spider
Network with Spider	Engine for Win & .NET
Publish (portable media)	Engine for Linux
Document filters also available for separate licensing	

Ask about fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991
www.dtSearch.com 1-800-IT-FINDS

msdn

magazine

AUGUST 2014 VOLUME 29 NUMBER 8

MOHAMMAD AL-SABT Editorial Director/mmeditor@microsoft.com

KENT SHARKEY Site Manager

MICHAEL DESMOND Editor in Chief/mmeditor@microsoft.com

LAKE LOW Features Editor

SHARON TERDEMAN Features Editor

DAVID RAMEL Technical Editor

WENDY HERNANDEZ Group Managing Editor

SCOTT SHULTZ Creative Director

JOSHUA GOULD Art Director

ALAN TAO Senior Graphic Designer

SENIOR CONTRIBUTING EDITOR Dr. James McCaffrey

CONTRIBUTING EDITORS Rachel Appel, Dino Esposito, Kenny Kerr, Julie Lerman, Ted Neward, Charles Petzold, David S. Platt, Bruno Terkaly, Ricardo Villalobos

Redmond Media Group

Henry Allain President, Redmond Media Group

Michele Imgrund Vice President, Lead Services Division

Tracy Cook Director, Client Services & Webinar Production

Irene Fincher Director, Audience Development & Lead Generation Marketing

ADVERTISING SALES: 818-674-3416/dlbianca@1105media.com

Dan LaBianca Chief Revenue Officer

Chris Kourtoglou Regional Sales Manager

Danna Vedder Regional Sales Manager/Microsoft Account Manager

David Seymour Director, Print & Online Production

Anna Lyn Bayaua Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA

Neal Vitale President & Chief Executive Officer

Richard Vitale Senior Vice President & Chief Financial Officer

Michael J. Valenti Executive Vice President

Erik A. Lindgren Vice President, Information Technology & Application Development

David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

MSDN Magazine (ISSN 1528-4859) is published monthly by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. \$35.00, International \$60.00. Annual digital subscription rates payable in U.S. funds are: U.S. \$25.00, International \$25.00. Single copies/back issues: U.S. \$10, all others \$12. Send orders with payment to: *MSDN Magazine*, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No. 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o *MSDN Magazine*, 4 Venture, Suite 150, Irvine, CA 92618.

Legal Disclaimer: The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@1105media.com

Reprints: For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International, Phone: 212-221-9595, E-mail: 1105reprints@parsintl.com, www.magreprints.com/QuickQuote.asp

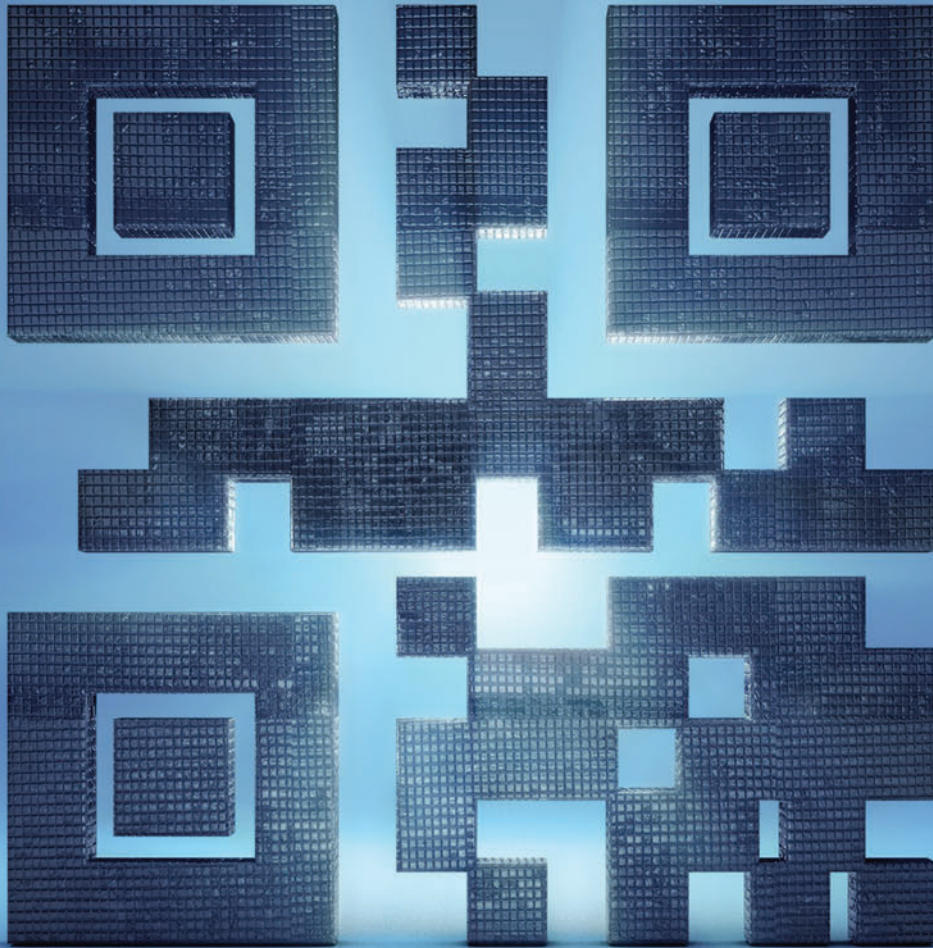
List Rental: This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Jane Long, Merit Direct. Phone: 913-685-1301; E-mail: jlong@meritdirect.com; Web: www.meritdirect.com/1105

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.



Printed in the USA

THE PREMIER BARCODE SDK



LEADTOOLS Barcode SDK is the world's premier toolkit for developing applications that require reading and writing **1D** and **2D barcodes**. With comprehensive support of over 100 different barcode types and sub-types such as UPC, EAN, Code 128, QR Code, Data Matrix, and PDF417, LEADTOOLS is the right choice.

.NET Windows API WinRT Linux iOS OS X Android HTML5/JavaScript





Everything I Need to Know I Learned from Bloom County

In my March column I wrote about the valuable lessons that the comic strip Calvin and Hobbes has to offer working developers (msdn.microsoft.com/magazine/dn605870). But Bill Watterson's tale of a 6-year-old boy and his imaginary tiger isn't the only font of funny-page wisdom, and I soon came to appreciate the technical insights of another of my all-time favorite comic strips—Berkeley Breathed's 1980s-era comic, Bloom County.

Bloom County hasn't aged nearly as well as Calvin and Hobbes. Breathed's comic strip bent to the political and was steeped in the issues of the day, replete with cracks about the Cold War, Abscam and Gary Hart. And where Calvin and Hobbes was both genuine and whimsical in its keenly observational way, Bloom County embraced the absurd. Talking penguins and schoolboy journalists rubbed elbows with gin-soaked senators and whatever monster wandered out of Binkley's bedroom closet of anxieties each night. And don't even get me started on Bill the Cat.

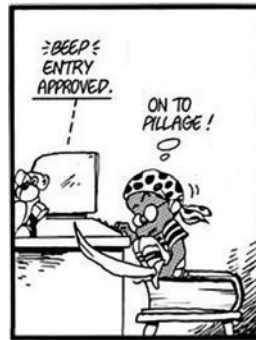
Bloom County at times played out like an absurdist's fever dream, and its depiction of humor around computers and technology was often unobvious. Still, beyond the hackneyed cracks about sentient computers and jabs at Apple, there were moments of insight. Most of these came courtesy of the character Oliver Wendell Jones, a 10-year-old boy who is a gifted programmer and inventor. Like Calvin, whose struggles with his Transmogrifier and other inventions shed light on good development practices, Oliver offers some valuable lessons of his own.



For instance, in one strip Oliver predicted the unintentional hilarity of machine translation, when he hacked into the publishing system of Pravda, the leading state-run newspaper of the Soviet Union. Hoping to spark a peace movement, Oliver and his friend Milo changed the day's Pravda headline to read, "Gorbachev Urges Disarmament: Total! Unilateral!" Alas, Oliver's translation in Russian actually read, "Gorbachev Sings Tractors: Turnip! Buttocks!"

My guess is that Oliver today works on the team at Google Translate. But his experience illuminates a legitimate point: Localization is hard. The language needs to be right, program interfaces must respect cultural expectations, and the entire package has to transition efficiently across geographies to contain cost and speed delivery.

But Oliver's enduring message to the development community is all about security—or the general lack of it in corporate and government computer systems. When Oliver broke into the Bell Telephone account system, he illustrated the fundamental



weakness—back in 1983—of single-factor authentication. It would seem absurd that a 10-year-old kid could walk into a secured corporate database, but consider the real-life case of a Montreal-area fifth-grader, who pled guilty to three charges stemming from attacks on government and police Web sites and databases in 2012 (bit.ly/1pHK40H).

Breathed's comic strip even predicted the risk posed by identity theft, when Oliver hacked into the IRS database and deleted all traces of his father from the system, causing him to wink out of existence. As Oliver said at the time, "Even the breathtaking political, philosophical and religious implications of this are dwarfed by the breathtaking implications of explaining this to Mom."

In fact, the ones who really needed this explained to them were the businesses, governments and financial institutions that, two decades later, would find themselves contending with increasing identity theft.

Berkeley Breathed's Bloom County was very much a product of the 1980s, and its humor and observations reflect the era. Though some of the jokes may not have aged well, the technical lessons taught by young Oliver Wendell Jones certainly have.

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for MSDN Magazine? Send them to the editor: mmeditor@microsoft.com.

© 2014 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system MSDN Magazine or any part of MSDN Magazine. If you have purchased or have otherwise properly acquired a copy of MSDN Magazine in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of MSDN Magazine (or any part of MSDN Magazine) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in MSDN Magazine are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. MSDN Magazine, MSDN, and Microsoft logos are used by 1105 Media, Inc. under license from owner.

Data Quality Tools for Developers



A better way to build in data verification

Since 1985, Melissa Data has provided the tools developers need to enhance databases with clean, correct, and current contact data. Our powerful, yet affordable APIs and Cloud services provide maximum flexibility and ease of integration across industry-standard technologies, including .NET, Java, C, and C++. Build in a solid framework for data quality and protect your investments in data warehousing, business intelligence, and CRM.

- Verify international addresses for over 240 countries
- Enhance contact data with phone numbers and geocodes
- Find, match, and eliminate duplicate records
- Sample source code for rapid application development
- Free trials with 120-day ROI guarantee



Address
Verification



Phone
Verification



Email
Verification



Geocoding



Matching/
Dedupe



Change of
Address

Melissa Data.

Architecting data quality success.

MELISSA DATA®

www.MelissaData.com 1-800-MELISSA



ASPOSE.TOTAL

Powerful APIs which enable developers to harness the complexity of file format processing within their apps.

Your File Format APIs



Aspose.Words

DOC, DOCX, RTF, HTML, PDF, XPS & other document formats.



Aspose.BarCode

JPG, PNG, BMP, GIF, TIFF, WMF, ICON & other image formats.



Aspose.Pdf

PDF, XML, XLS-FO, HTML, BMP, JPG, PNG & other image formats.



Aspose.Imaging

PDF, BMP, JPG, GIF, TIFF, PNG, PSD & other image formats.



Aspose.Cells

XLS, XLSX, XLSM, XLTX, CSV, SpreadsheetML & image formats.



Aspose.Tasks

XML, MPP, SVG, PDF, TIFF, PNG, CSV, MPT & other formats.



Aspose.Slides

PPT, PPTX, POT, POTX, XPS, HTML, PNG, PDF & other formats.



Aspose.Diagram

VSD, VSDX, VSS, VST, VSX & other formats.



Aspose.Email

MSG, EML, PST, EMLX & other formats.



Aspose.Note

ONE, PNG, JPG, BMP, GIF & PDF.

... and more!

100% Standalone - No Office Automation



.NET Libraries



Java Libraries



Cloud APIs



Android Libraries



Scan for a 20% saving!



US: +1 888 277 6734
sales@aspose.com

EU: +44 141 416 1112
sales.europe@aspose.com

AU: +61 2 8003 5926
sales.asiapacific@aspose.com



GROUPDOCS.TOTAL

Professional APIs that allow developers to empower their apps with document collaboration capabilities.

Your Document Collaboration APIs



GroupDocs.Viewer

Native-text, high-fidelity HTML5 document viewer with support for over 49 file formats.



GroupDocs.Signature

Electronic signature API that gives your apps legally binding e-signature capabilities.



GroupDocs.Conversion

Universal document converter for fast conversion between more than 49 file formats.



GroupDocs.Annotation

A powerful API that lets developers annotate Microsoft Office, PDF and other documents within their own apps.



GroupDocs.Assembly

Incorporates data entered by users through online forms into both Microsoft Office and PDF documents.



GroupDocs.Comparison

A diff view API that allows end users to quickly find differences between two revisions of a document.

100% Standalone - No Office Automation



.NET Libraries



Java Libraries



Cloud APIs



Cloud Apps

SALES INQUIRIES: +1 214 329 9760

sales@groupdocs.com



GROUPDOCS

Your Document Collaboration APIs

www.groupdocs.com



Documents, Databases and Eventual Consistency

Let's say you've spent the last couple of years stuck in a cave with no Internet connectivity. Now that you're back in the real world, you're assigned to a brand-new project. At the first technical meeting, you listen to an animated discussion between team members with strong opinions about document databases.

What the heck is a document database? Didn't we used to store data in a plain old relational SQL Server instance? What benefits can document databases bring to the table? You diligently listen to the various opinions while trying to make sense of the whole thing. You wonder how to realistically answer whether you should consider using document databases instead of relational storage for your next project.

This article doesn't intend to take a definitive stance regarding document databases and NoSQL stores in general. However, I hope to illustrate a skeptical perspective of someone who's always ready to explore better ways of doing things and always looking for concrete and measurable benefits, but not at all costs.

Beyond SQL

The relational model and Structured Query Language (SQL) have been around for more than 40 years. That's an incredible amount of time for this rapidly changing industry. Over the past few decades, the relational model has fended off attacks from object-oriented databases that seemed ready to overtake the old-fashioned relational model on the wave of object-oriented languages and modeling patterns.

That never happened, though. What did happen was the emergence of object-relational mapping (ORM) tools. In the .NET space, NHibernate initially arose as a de facto standard and was more recently equaled by the Entity Framework. There have also been other similar commercial and open source frameworks from a variety of vendors and contributors. So the first point is that object modeling is not a strong enough reason to push the relational model to the corner.

Yet more companies are using NoSQL stores. A good question to ask is, "Where are NoSQL stores being used?" This is a much better question than, "How can I take advantage of NoSQL stores?" Examining realistic technology use cases to see if they match your own is preferable over blindly trying to find reasons to use a given technology. If you explore the context of using NoSQL stores, you recognize the following common aspects:

- Large—often, unpredictably large—volumes of data and possibly millions of users
- Thousands of queries per second

- Presence of unstructured/semi-structured data that might come in different forms, but still need the same treatment (polymorphic data)
- Cloud computing and virtual hardware for extreme scalability

If your project doesn't match any of these conditions, you can hardly expect to find NoSQL particularly rewarding. Using NoSQL outside of such conditions may just end up being a different way of doing the same old things.

Structural Differences

The need to build highly interactive applications that push a huge number of writes and reads to the database server is well-suited for NoSQL. In a relational model, relationships between tables work great as long as all the tables involved have fixed schemas. All involved tables must also provide a faithful and realistic representation of the domain model.

If your data—regardless of size and access frequency—fits nicely into "structured tables," then you can be reasonably certain that good old SQL Server will work effectively. Classic SQL technology is far from dead, and improves over time. The column store feature in SQL Server 2014 helps you handle hundreds of columns. Having hundreds of columns may affect query performance. A huge number of columns is also sometimes the result of attempting to map semi-structured data to a relational model.

A column store is a plain table with the usual set of rows and columns, except the content is physically laid out and stored by column. As a result, all data in any given column is stored on the same SQL physical page. If you need to query all data from a selected large number of columns, this new form of storage can deliver a significant performance boost without re-architecting the persistence layer.

A growing number of today's applications deal with more intricate and complex data models that hardly fit in structured tables. Subsequently, application architects wonder if workarounds and compromises are really necessary for storing and querying such partially structured data within the boundaries of rigid SQL schemas. If you can successfully model your data to a relational schema, then you likely incur in JOIN statements for a large number of common queries. So when tables grow beyond imagination, query performance inevitably decreases. As this will happen when a lot of customers are using the application, delayed response can affect the business behind the application.

You have two other options from which to choose—SQL and NoSQL. Each sounds like it's replacing the other. SQL and NoSQL aren't different flavors of the same technology. Both deal with persistence of data, but with a number of structural differences.

NoSQL databases don't use fixed data schemas and relationships. As such, they don't dictate a model. NoSQL databases free you from modeling the domain space. You can persist resulting objects in logical groups. Design a domain model, work with a graph of objects and the NoSQL store just handles object serialization to disk.

Working with Documents?

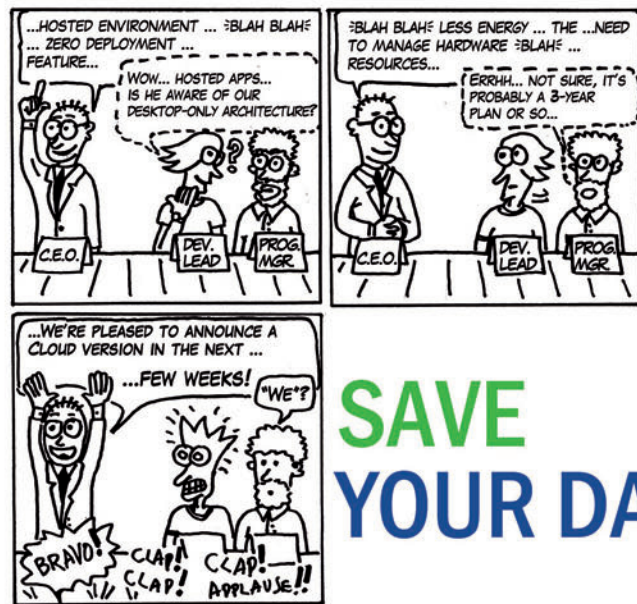
In this case, the term “document” is roughly equivalent to the term “record.” Collections of documents may recall tables of records. The key difference is that each object in a collection may have a different schema than all other objects in the same collection. Yet all the documents are logically related.

This distinction is more subtle than it might first appear. Suppose you're managing the bios of book authors. You might not know the same amount of data for each author. Some data that describes Author1 may be different from data that describes Author2. If you intend the bio to be a collection of attributes, you actually have a fixed schema with a few optional columns. If you intend the bio to be a collection of files like a Word-based CV, an XML stream with the list of published books and comments, links to YouTube interviews, and a few attributes such as vital statistics, then the schema is much less defined and hardly fits within the rigid boundaries of a SQL store.

The level of polymorphism in your data is an excellent metric by which to measure how much NoSQL can help. Suppose you're building a system in accordance with event-sourcing architecture. In an event-sourcing architecture, the persistence model of the application is the plain history of events. Every user action originates one or more events on the server side. Recording the event is all you need to keep track of the application state.

In an e-commerce scenario, for example, when the user submits an order, it starts a workflow. This might generate a number of domain events—order-submitted, order-validated, order-denied, order-created, order-being-processed, order-shipping, order-shipped, order-returned, order-updated and so on. These events all relate to the same order. Processing the sequence of events lets you build and rebuild the current state of the order.

Dealing with orders in a realistic e-commerce system isn't just a matter of maintaining an Orders table with an updated Status column. It means tracking all actions performed on the data that represents an order. These actions may be quite different and involve different data. For example, the order-returned event might have nearly no associated data. The order-submitted event likely carries the entire set of information coming from the user. The event order-updated likely contains only the variation made to the existing order.



SAVE YOUR DAY!



SUPPORT FOR
VISUAL STUDIO
2013

Use CodeFluent Entities

CodeFluent Entities is a unique product integrated into Visual Studio that allows you to generate database scripts, code (C#, VB), web services and UIs.

"I recently spent a week attending a course on Entity Framework but CodeFluent Entities provides so much more and is decidedly easier to understand and implement"

Peter Stanford - Artefaction - Australia

* Source : <http://visualstudiogallery.msdn.microsoft.com/B6299BBF-1EF1-436D-B618-66E8C16AB410>

To get a license worth \$399 for free
Go to www.softfluent.com/forms/msdn-2014

 **CodeFluent Entities**
tools for developers, by developers

More information: www.softfluent.com Contact us: info@softfluent.com

Your order will have more of a document feel, as it's fully described through a list of highly heterogeneous events. Each event is just an object to store. Most e-commerce systems probably use a relational store to address these problems. A NoSQL approach to storing this list of events could be quite interesting and promising. To learn more about event-sourcing architecture, download the free e-book, "Exploring CQRS and Event Sourcing," from the Microsoft Download Center at bit.ly/1lesmzm.

Is Eventual Consistency an Issue?

Eventual consistency is another relevant structural difference between SQL and NoSQL. Even when you can easily recognize documents in your data model, the impact of eventual consistency on deployed applications is the true discriminant to the final decision.

Eventual consistency is when reads and writes aren't aligned to the same data. Most NoSQL systems are eventually consistent in the sense that they guarantee if no updates are made to a given object for a sufficient period of time, then a query returns what the last command has written.

In most cases, eventual consistency isn't an issue at all. You generally need to be as consistent as possible *within* the Bounded Context, but you don't really need any level of consistency *across* Bounded Contexts. As the system grows, despite the technology, you can't rely on consistency.

There's a simple test to see whether eventual consistency is an issue. How would you consider a scenario in which a command writes some data, but a successive read returns stale data? If it's absolutely crucial you're constantly able to read back what has just been written, then you have two options:

- Avoid NoSQL databases
- Configure the NoSQL database to be consistent

Consider the following code snippet that assumes the use of RavenDB—a popular .NET NoSQL database:

```
DocumentSession
    .Store(yourObject);
DocumentSession
    .Query<YourObjectType>()
    .Where(t => t.Id == id)
```

The first line stores an object to the RavenDB archive. The second line attempts to read the object back, making a query on the same store session for the value of some Id property on the saved object. With the default database configuration, what you'll read isn't the same as what you've just written.

As far as RavenDB is concerned, writing on the store and updating indexes used by the query engine are distinct operations. Index updates occur as scheduled operations. That misalignment doesn't last more than a few seconds if in the meantime there are no other updates to the same object. Here's a way to force actual consistency:

```
_instance.Conventions.DefaultQueryingConsistency =
    ConsistencyOptions.AlwaysWaitForNonStaleResultsAsOfLastWrite;
```

When you do so, though, that read doesn't return until the index has been updated. A trivial read, therefore, might take a few seconds to complete. From this, you see that NoSQL stores fill a niche in the industry. They do pose challenges, though. NoSQL addresses some architecture issues while neglecting others.

There are better ways to "wait" for indexes. Those generally depend on the scenario you're facing. Thus, it's better to decide the type of consistency a query needs at query time, as in the following example:

```
using( var session = store.OpenSession() )
{
    var query = session.Query<Person>()
        .Customize(c=> c.WaitForNonStaleResultsAsOfLastWrite() )
        .Where( p => /* condition */ );
}
```

Here, the `WaitForNonStaleResultsAsOfLastWrite` query customization is instructing the server to wait for the relevant index to have indexed the last document written and ignore eventual documents arriving at the server after the query has been issued.

This helps in certain scenarios with a high write ratio, where indexes are always stale. This is by design. There are many `WaitForNonStaleResultsXxxx` methods that have different behaviors and solve slightly different scenarios. Another possibility is to fully embrace eventual consistency and simply ask the server if the returned results are stale. Then behave accordingly:

```
using( var session = store.OpenSession() )
{
    RavenQueryStatistics stats;
    var query = session.Query<Person>()
        .Statistics( out stats )
        .Where( p => /* condition */ );
}
```

In this example, you're not waiting for indexes. You're asking the server to also output the query statistics that let you know if the returned results are stale. Given the scenario you're trying to resolve, you should have enough information to take the best possible decision.

Polyglot Persistence

At the end of the day, the point isn't to call relational stores dead and replace them with the NoSQL product of choice. The point is to understand the mechanics of the system and characteristics of the data and working out the best possible architecture. The most foreseeable concrete application for NoSQL stores is as an event store in the context of an event-sourcing architecture.

For systems where there isn't a simple yes or no answer to whether you should use relational stores, the best you can do is to consider polyglot persistence. Instead of forcing a choice between NoSQL and relational databases, you could look into a storage layer that combines the strengths of NoSQL and relational databases. This type of storage system tackles different problems in the most appropriate way.

Within the context of an enterprise, you should use different storage technologies to store different types of data. This is especially true in a service-oriented architecture. In this case, each service may have its own storage layer. There would be no reason to unify storage under a single technology or product. Polyglot persistence does require that you learn different storage technologies and products. As a training cost, though, it's a reasonable investment. ■

DINO ESPOSITO is the co-author of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2014) and "Programming ASP.NET MVC 5" (Microsoft Press, 2014). A technical evangelist for the .NET Framework and Android platforms at JetBrains and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents.wordpress.com and on Twitter at twitter.com/despos.

THANKS to the following technical expert for reviewing this article:
Mauro Servienti (Managed Designs)



GdPicture.NET 10



GdPicture.NET 10, the Ultimate Imaging SDK

Q&A with Loïc Carrère, Creator of GdPicture.NET,
Owner and CEO of ORPALIS

Q How would you describe your flagship product, GdPicture.NET, in 2 words?

A I would say: complete and customized.

1. Complete

Starting with document acquisition (covering both TWAIN and WIA protocols), document imaging and image processing, over the years we've added barcoding features (linear and 2D), OCR, PDF viewing and processing, then annotations, forms processing and JBIG2/JPEG 2000 compression. With version 10 we've expanded our offer with DICOM support for medical imaging and a MICR reader perfect for banking solutions. We also developed a very innovative color detection plugin for quality vs. size optimization of image files. We can now say that we're covering pretty much all domains of document imaging.

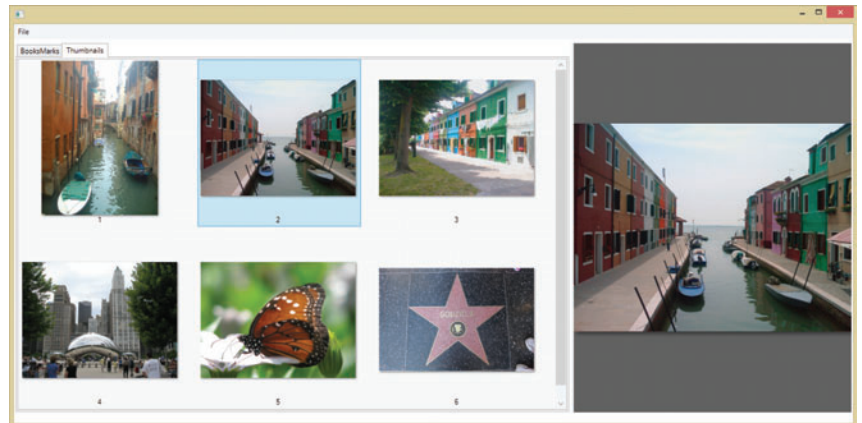
2. Customized

While working on the upcoming major release of GdPicture.NET we decided to take the word "custom" to the next level. Since the first version of GdPicture we've listened to our customers and we've implemented numerous functionalities on demand. We also offer dedicated A-to-Z custom development projects.

Whether companies need specific imaging applications to be integrated in their solutions or want to get a fully developed software for end-users, we'll be offering our services.

Q How do developers typically use GdPicture.NET controls?

A Our SDK is perfect for any developer building applications for Microsoft operating systems or for the Web. Whether they are using ASP.NET, WPF or any environment based on WinForms such as vb6, .NET, vfp...



Q What makes ORPALIS different from other companies? In other word, what is the philosophy at ORPALIS?

A GdPicture has been on the market for more than 10 years now. Our experience in the field is renowned worldwide. Our technologies have been recognized and validated by the Ministry of Research of France and we're now labelled as "Young Innovative Company." It's a great reward for our work and it brings us a solid scientific backing.

But the greatest reward of all is probably to see our clients' businesses grow thanks to their solutions based on GdPicture.NET. Since the beginning we decided to provide royalty-free solutions. We want to encourage both companies and free-lancers to launch their projects for a minimum cost with our versatile and all-inclusive SDK. Seeing them succeed, that's what we strive for.

Q What's next?

A ORPALIS is a fast-growing company: our priorities for 2015 is to attract new talents to expand our team, our knowledge and as a consequence, our offer.

To learn more please visit our website →

www.gdpicture.com



DirectComposition: A Retained-Mode API to Rule Them All

Graphics APIs tend to fall into one of two very different camps. There are the immediate-mode APIs with well-known examples, including Direct2D and Direct3D. Then there are the retained-mode APIs such as Windows Presentation Foundation (WPF) or any XAML or declarative API. Modern browsers offer a clear distinction between the two graphics modes, with Scalable Vector Graphics providing a retained-mode API and the Canvas element providing an immediate-mode API.

Retained-mode assumes the graphics API will retain some representation of the scene, such as a graph or tree of objects, which can then be manipulated over time. This is convenient and simplifies the development of interactive applications. In contrast, an immediate-mode API doesn't include a built-in scene graph and instead relies on the application to construct the scene using a sequence of drawing commands. This has tremendous performance benefits. An immediate-mode API such as Direct2D will typically buffer vertex data from multiple geometries, coalesce a large number of drawing commands and much more. This is particularly beneficial with a text-rendering pipeline, as glyphs first need to be written to a texture and down-sampled before clear-type filtering is applied and the text makes its way to the render target. This is one of the reasons why many other graphics APIs, and increasingly many third-party applications, now rely on Direct2D and DirectWrite for text rendering.

The choice between immediate-mode and retained-mode traditionally came down to a trade-off between performance and productivity. Developers could pick the Direct2D immediate-mode API for absolute performance or the WPF retained-mode API for productivity or convenience. DirectComposition changes this equation by allowing developers to blend the two far more naturally. It blurs the line between immediate-mode and retained-mode APIs because it provides a retained-mode for graphics, but without imposing any memory or performance overhead. It achieves this feat by focusing on bitmap composition rather than attempting to compete with other graphics APIs. DirectComposition simply provides the visual tree and the composition infrastructure such that bitmaps rendered with other technologies can be easily manipulated and composed together. And unlike WPF, DirectComposition is an integral part of the OS graphics infrastructure and avoids all of the performance and airspace issues that have traditionally plagued WPF applications.

If you've read my two previous columns on DirectComposition (msdn.microsoft.com/magazine/dn745861 and msdn.microsoft.com/magazine/dn786854), you should already have a sense of what the composition engine is capable of. Now I want to make that a lot more explicit by showing you how you can use DirectComposition to manipulate visuals drawn with Direct2D in a way that's very appealing to developers accustomed to retained-mode APIs. I'm going to show you how to create a simple window that presents circles as "objects" that can be created and moved around, with full support for hit testing and changes to Z-order. You can see what this might look like in the example in **Figure 1**.

Graphics APIs tend to fall into one of two very different camps.

Although the circles in **Figure 1** are drawn with Direct2D, the application draws a circle only once to a composition surface. This composition surface is then shared among the composition visuals in a visual tree bound to the window. Each visual defines an offset relative to the window at which its content—the composition surface—is positioned and ultimately rendered by the composition engine. The user can create new circles and move them around with a mouse, pen or finger. Every time a circle is selected, it moves to the top of the Z-order so it appears above any other circles in the window. While I certainly don't need a retained-mode API to achieve such a simple effect, it does

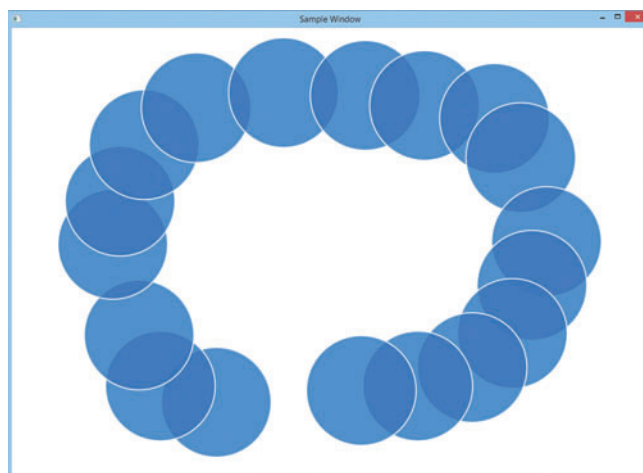


Figure 1 Dragging Circles Around

Code download available at msdn.microsoft.com/magazine/msdnmag0814.

serve as a good example of how the DirectComposition API works along with Direct2D to achieve some powerful visual effects. The goal is to build an interactive application whose WM_PAINT handler isn't responsible for keeping the window's pixels up-to-date.

I'll start with a new SampleWindow class that derives from the Window class template I introduced in my previous column. The Window class template just simplifies message dispatching in C++:

```
struct SampleWindow : Window<SampleWindow>
{
};
```

As with any modern Windows application, I need to handle dynamic DPI scaling so I'll add two floating-point members to keep track of the DPI scaling factors for the X and Y axis:

```
float m_dpiX = 0.0f;
float m_dpiY = 0.0f;
```

You can initialize these on demand, as I illustrated in my previous column, or within your WM_CREATE message handler. Either way, you need to call the MonitorFromWindow function to determine the monitor that has the largest area intersecting the new window. Then you simply call the GetDpiForMonitor function to retrieve its effective DPI values. I've illustrated this a number of times in previous columns and courses so I won't reiterate it here.

I'll use a Direct2D ellipse geometry object to describe the circle to be drawn so I can later use this same geometry object for hit testing. While it's more efficient to draw a D2D1_ELLIPSE structure than a geometry object, the geometry object provides hit testing and the drawing will be retained. I'll keep track of both the Direct2D factory and the ellipse geometry:

```
ComPtr<ID2D1Factory> m_factory;
ComPtr<ID2D1EllipseGeometry> m_geometry;
```

PhysicalToLogical is a helper function I routinely use for DPI scaling when combining APIs that have differing levels of support for DPI scaling (or none at all).

In my previous column I showed you how to create a Direct2D device object directly with the D2D1CreateDevice function rather than by using a Direct2D factory. This is certainly an acceptable way to continue, but there's a catch. Direct2D factory resources, while they are device-independent and need not be recreated when device loss occurs, can be used only with Direct2D devices created by the same Direct2D factory. Because I want to create the ellipse geometry up front, I need a Direct2D factory object to create it. I could, perhaps, wait until I've created the Direct2D device with the D2D1CreateDevice function and then retrieve the underlying factory with the GetFactory method, and then use that factory object to create the geometry, but that seems rather contrived. Instead, I'll just create a Direct2D factory and use it to create both the ellipse geometry and the device object as needed. **Figure 2** illustrates how to create the Direct2D factory and geometry objects.

Figure 2 Creating the Direct2D Factory and Geometry Objects

```
void CreateFactoryAndGeometry()
{
    D2D1_FACTORY_OPTIONS options = {};

    #ifdef _DEBUG
    options.debugLevel = D2D1_DEBUG_LEVEL_INFORMATION;
    #endif

    HR(D2D1CreateFactory(D2D1_FACTORY_TYPE_SINGLE_THREADED,
        options,
        m_factory.GetAddressOf()));

    D2D1_ELLIPSE const ellipse = Ellipse(Point2F(50.0f, 50.0f),
        49.0f,
        49.0f);

    HR(m_factory->CreateEllipseGeometry(ellipse,
        m_geometry.GetAddressOf()));
}
```

The CreateFactoryAndGeometry method can then be called by the SampleWindow's constructor to prepare these device-independent resources. As you can see, the ellipse is defined around a center point 50 pixels along the X and Y axis, as well as a radius of 49 pixels for both the X and Y radius, making this ellipse into a circle. I'll be creating a 100x100 composition surface. I've chosen a radius of 49 pixels because the default stroke drawn by Direct2D straddles the perimeter and it would otherwise be clipped.

Next up are the device-specific resources. I need a backing Direct3D device, a composition device to commit changes to the visual tree, a composition target to keep the visual tree alive, a root visual that will represent the parent of all of the circle visuals, and a shared composition surface:

```
ComPtr<ID3D11Device> m_device3D;
ComPtr<IDCompositionDesktopDevice> m_device;
ComPtr<IDCompositionTarget> m_target;
ComPtr<IDCompositionVisual2> m_rootVisual;
ComPtr<IDCompositionSurface> m_surface;
```

I've introduced these various objects in my earlier DirectX articles and, in particular, in my two previous columns on DirectComposition. I also discussed and illustrated how you should handle device creation and loss so I won't repeat that here. I'll just call out the CreateDevice2D method that needs to be updated to use the previously created Direct2D factory:

```
ComPtr<ID2D1Device> CreateDevice2D()
{
    ComPtr<IDXGIDevice3> deviceX;
    HR(m_device3D.As(&deviceX));
    ComPtr<ID2D1Device> device2D;

    HR(m_factory->CreateDevice(deviceX.Get(), device2D.GetAddressOf()));

    return device2D;
}
```

Now I'll create the shared surface. I need to be careful to use the ComPtr class template's ReleaseAndGetAddressOf method to ensure the surface can be safely recreated after device loss or due to changes in DPI scaling. I also need to be careful to preserve the logical coordinate system that my application is using while translating the dimensions into physical pixels for the DirectComposition API:

```
HR(m_device->CreateSurface(
    static_cast<unsigned>(LogicalToPhysical(100, m_dpiX)),
    static_cast<unsigned>(LogicalToPhysical(100, m_dpiY)),
    DXGI_FORMAT_B8G8R8A8_UNORM,
    DXGI_ALPHA_MODE_PREMULTIPLIED,
    m_surface.ReleaseAndGetAddressOf()));
```

I can then call the composition surface's `BeginDraw` method to receive a `Direct2D` device context with which to buffer drawing commands:

```
HR(m_surface->BeginDraw(
    nullptr,
    __uuidof(dc),
    reinterpret_cast<void **>(dc.GetAddressOf()),
    &offset));
```

And then I need to tell `Direct2D` how to scale any drawing commands:

```
dc->SetDpi(m_dpiX,
    m_dpiY);
```

And I need to transform the output to the offset provided by `DirectComposition`:

```
dc->SetTransform(Matrix3x2F::Translation(PhysicalToLogical(offset.x, m_dpiX),
    PhysicalToLogical(offset.y, m_dpiY)));
```

`PhysicalToLogical` is a helper function I routinely use for DPI scaling when combining APIs that have differing levels of support for DPI scaling (or none at all). You can see the `PhysicalToLogical` function and the corresponding `LogicalToPhysical` function in **Figure 3**.

Now I can simply draw a blue circle with a solid color brush created for just this purpose:

```
ComPtr<ID2D1SolidColorBrush> brush;

D2D1_COLOR_F const color = ColorF(0.0f, 0.5f, 1.0f, 0.8f);

HR(dc->CreateSolidColorBrush(color,
    brush.GetAddressOf()));
```

Next, I must clear the render target before filling the ellipse geometry and then stroking or drawing its outline with the modified brush:

```
dc->Clear();

dc->FillGeometry(m_geometry.Get(),
    brush.Get());

brush->SetColor(ColorF(1.0f, 1.0f, 1.0f));

dc->DrawGeometry(m_geometry.Get(),
    brush.Get());
```

Finally, I must call the `EndDraw` method to indicate the surface is ready for composition:

```
HR(m_surface->EndDraw());
```

Now it's time to create circles. In my previous columns I created only a single root visual, but this application needs to create visuals on demand, so I'll just wrap that up in a convenient helper method:

```
ComPtr<IDCompositionVisual2> CreateVisual()
{
    ComPtr<IDCompositionVisual2> visual;
    HR(m_device->CreateVisual(&visual));
    return visual;
}
```

One of the interesting aspects of the `DirectComposition` API is that it's effectively a write-only interface to the composition engine. While it retains a visual tree for your window, it doesn't provide any

Figure 3 Converting Between Logical and Physical Pixels

```
template <typename T>
static float PhysicalToLogical(T const pixel,
    float const dpi)
{
    return pixel * 96.0f / dpi;
}

template <typename T>
static float LogicalToPhysical(T const pixel,
    float const dpi)
{
    return pixel * dpi / 96.0f;
}
```

getters you can use to interrogate the visual tree. Any information, such as a visual's position or Z-order, must be retained directly by the application. This avoids unnecessary memory overhead and also avoids potential race conditions between the application's view of the world and the composition engine's transactional state. So I'll go ahead and create a `Circle` structure to keep track of each circle's position:

```
struct Circle
{
    ComPtr<IDCompositionVisual2> Visual;
    float LogicalX = 0.0f;
    float LogicalY = 0.0f;
};
```

The composition visual effectively represents the circle's setters while the `LogicalX` and `LogicalY` fields are the getters. I can set the visual's position with the `IDCompositionVisual2` interface and I can retain and later retrieve its position with the other fields. This is necessary both for hit testing and for restoring the circles after device loss. To avoid these getting out of sync, I'll simply provide a helper method for updating the visual object based on the logical position. The `DirectComposition` API has no idea how the content might be positioned and scaled, so I need to make the necessary DPI calculations myself:

```
void UpdateVisualOffset(float const dpiX,
    float const dpiY)
{
    HR(Visual->SetOffsetX(LogicalToPhysical(LogicalX, dpiX)));
    HR(Visual->SetOffsetY(LogicalToPhysical(LogicalY, dpiY)));
}
```

I'll add another helper method for actually setting the circle's logical offset. This one relies on `UpdateVisualOffset` to ensure that the `Circle` structure and the visual object are in sync:

```
void SetLogicalOffset(float const logicalX,
    float const logicalY,
    float const dpiX,
    float const dpiY)
{
    LogicalX = logicalX;
    LogicalY = logicalY;

    UpdateVisualOffset(dpiX,
        dpiY);
}
```

Finally, as circles are added to the application, I'll need a simple constructor to initialize the structure, taking ownership of an `IDCompositionVisual2` reference:

```
Circle(ComPtr<IDCompositionVisual2> && visual,
    float const logicalX,
    float const logicalY,
    float const dpiX,
    float const dpiY) :
    Visual(move(visual))
{
    SetLogicalOffset(logicalX,
        logicalY,
        dpiX,
        dpiY);
}
```

I can now keep track of all the application's circles with a standard list container:

```
list<Circle> m_circles;
```

While I'm here I'll also add a member to track any selected circle:

```
Circle * m_selected = nullptr;
float m_mouseX = 0.0f;
float m_mouseY = 0.0f;
```

The mouse offset will also help to produce natural movement. I'll get the housekeeping out of the way before I look at the actual mouse interaction that will ultimately create the circles and allow me to

Switch to Amyuni PDF

Create & Edit PDFs in .Net - ActiveX - WinRT

- Edit, process and print PDF 1.7 documents
- Create, fill-out and annotate PDF forms
- Fast and lightweight 32- and 64-bit components for .Net and ActiveX/COM
- New WinRT Component enables publishing C#, C++CX or Javascript apps to Windows Store
- New Postscript/EPS to PDF conversion module



Complete Suite of Accurate PDF Components

- All your PDF processing, conversion and editing in a single package
- Combines Amyuni PDF Converter and PDF Creator for easy licensing, integration and deployment
- Includes our Microsoft WHQL certified PDF Converter printer driver
- Export PDF documents into other formats such as JPeg, PNG, XAML or HTML5
- Import and Export XPS files using any programming environment



Advanced HTML to PDF & XAML

- Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- Easy Integration and deployment within developer's applications
- WebkitPDF is based on the Webkit Open Source library and Amyuni PDF Creator



High Performance PDF Printer For Desktops and Servers



- Our high-performance printer driver optimized for Web, Application and Print Servers. Print to PDF in a fraction of the time needed with other tools. WHQL tested for Windows 32 and 64-bit including Windows Server 2012 R2 and Windows 8.1
- Standard PDF features included with a number of unique features. Interface with any .Net or ActiveX programming language
- Easy licensing and deployment to fit system administrator's requirements



AMYUNI

All development tools available at

www.amyuni.com

USA and Canada

Toll Free: 1866 926 9864
Support: 514 868 9227
sales@amyuni.com

Europe

UK: 0800-015-4682
Germany: 0800-183-0923
France: 0800-911-248

move them around. The `CreateDeviceResources` method needs to recreate any visual objects, should device loss occur, based on any previously created circles. It wouldn't do if the circles disappear. So right after creating or recreating the root visual and the shared surface, I'll iterate over this list, create new visual objects and reposition them to match the existing state. **Figure 4** illustrates how this all comes together using what I've already established.

The other bit of housekeeping has to do with DPI scaling. The composition surface that contains the circle's pixels as rendered by Direct2D must be recreated to scale, and the visuals themselves must also be repositioned so their offsets are proportional to one another and to the owning window. The `WM_DPICHANGED` message handler first recreates the composition surface—with the help of the `CreateDeviceScaleResources` method—and then updates the content and position for each of the circles:

```
if (!IsDeviceCreated()) return;
CreateDeviceScaleResources();

for (Circle & circle : m_circles)
{
    HR(circle.Visual->SetContent(m_surface.Get()));
    circle.UpdateVisualOffset(m_dpiX, m_dpiY);
}
```

```
HR(m_device->Commit());
```

Now I'll deal with the pointer interaction. I'll let the user create new circles if the left mouse button is clicked while the `Control` key is pressed. The `WM_LBUTTONDOWN` message handler looks something like this:

```
if (wparam & MK_CONTROL)
{
    // Create new circle
}
else
{
    // Look for existing circle
}
```

```
HR(m_device->Commit());
```

Assuming a new circle needs to be created, I'll start by creating a new visual and setting the shared content before adding it as a child of the root visual:

```
ComPtr<IDCompositionVisual2> visual = CreateVisual();
HR(visual->SetContent(m_surface.Get()));
HR(m_rootVisual->AddVisual(visual.Get(), false, nullptr));
```

The new visual is added in front of any existing visuals. That's the `AddVisual` method's second parameter at work. If I had set this to `true` then the new visual would've been placed at the back of any existing siblings. Next, I need to add a `Circle` structure to the list so I can later support hit testing, device loss and DPI scaling:

```
m_circles.emplace_front(move(visual),
    PhysicalToLogical(LOWORD(lparam), m_dpiX) - 50.0f,
    PhysicalToLogical(HIWORD(lparam), m_dpiY) - 50.0f,
    m_dpiX,
    m_dpiY);
```

I'm careful to place the newly created `Circle` at the front of the list so I can naturally support hit testing in the same order the visual tree implies. I also initially position the visual so it's centered on the mouse position. Finally, assuming the user doesn't immediately release the mouse, I also capture the mouse and keep track of which circle will potentially be moved around:

```
SetCapture(m_window);
m_selected = &m_circles.front();
m_mouseX = 50.0f;
m_mouseY = 50.0f;
```

The mouse offset allows me to smoothly drag any circle regardless of where on the circle the mouse pointer initially goes down. Looking for an existing circle is a little more involved. Here, again, I need to manually apply DPI awareness. Fortunately, Direct2D makes this a breeze. First, I need to iterate over the circles in the natural Z-order. Happily, I already placed new circles at the front of the list so this is a simple matter of iterating from beginning to end:

```
for (auto circle = begin(m_circles); circle != end(m_circles); ++circle)
{
}
```

The trick is that the geometry object provides only the shape of the circle and not its position.

I'm not using a range-based `for` statement because it's going to be more convenient to actually have iterators handy in this case. Now where are the circles? Well, each circle keeps track of its logical position relative to the top-left corner of the window. The mouse message's `LPARAM` also contains the pointer's physical position relative to the top-left corner of the window. But it's not enough to translate them to a common coordinate system because the shape I need to hit test for isn't a simple rectangle. The shape is defined by a geometry object and Direct2D provides the `FillContainsPoint` method to implement hit testing. The trick is that the geometry object provides only the shape of the circle and not its position. For hit testing to work effectively, I'll need to first translate the mouse position such that it's relative to the geometry object. That's easy enough:

```
D2D1_POINT_2F const point =
    Point2F(LOWORD(lparam) - LogicalToPhysical(circle->LogicalX, m_dpiX),
    HIWORD(lparam) - LogicalToPhysical(circle->LogicalY, m_dpiY));
```

Figure 4 Creating the Device Stack and Visual Tree

```
void CreateDeviceResources()
{
    ASSERT(!IsDeviceCreated());
    CreateDevice3D();
    ComPtr<ID2D1Device> const device2D = CreateDevice2D();

    HR(DCompositionCreateDevice2(
        device2D.Get(),
        __uuidof(m_device),
        reinterpret_cast<void **>(m_device.ReleaseAndGetAddressOf())));

    HR(m_device->CreateTargetForHwnd(m_window,
        true,
        m_target.ReleaseAndGetAddressOf()));

    m_rootVisual = CreateVisual();
    HR(m_target->SetRoot(m_rootVisual.Get()));
    CreateDeviceScaleResources();

    for (Circle & circle : m_circles)
    {
        circle.Visual = CreateVisual();
        HR(circle.Visual->SetContent(m_surface.Get()));
        HR(m_rootVisual->AddVisual(circle.Visual.Get(), false, nullptr));
        circle.UpdateVisualOffset(m_dpiX, m_dpiY);
    }

    HR(m_device->Commit());
}
```




Desktop. Web. Mobile. Your next great app starts here.

From interactive Desktop applications, to immersive Web and Mobile solutions, development tools built to meet your needs today and ensure your continued success tomorrow.

Download your free 30-day trial today and Experience the DevExpress Difference.

www.devexpress.com/try

DevExpress®

WIN ASP WPF SL VCL XAF CR

All trademarks or registered trademarks are property of their respective owners.

But I'm not quite ready to call the `FillContainsPoint` method. The other issue is that the geometry object doesn't know anything about the render target. When I used the geometry object to draw the circle, it was the render target that scaled the geometry to match the DPI values of the target. So I need a way to scale the geometry prior to performing hit testing so it will reflect the size of the circle, corresponding to what the user is actually seeing on screen. Once again, `Direct2D` comes to the rescue. `FillContainsPoint` accepts an optional 3x2 matrix to transform the geometry prior to testing whether the given point is contained within the shape. I can simply define a scale transform given the window's DPI values:

```
D2D1_MATRIX_3X2_F const transform = Matrix3x2F::Scale(m_dpiX / 96.0f,
                                                    m_dpiY / 96.0f);
```

As usual, changes to the visual tree must be committed for them to be realized.

The `FillContainsPoint` method will then tell me whether the point is contained within the circle:

```
BOOL contains = false;

HR(m_geometry->FillContainsPoint(point,
                                transform,
                                &contains));

if (contains)
{
    // Reorder and select circle

    break;
}
```

If the point is contained within the circle, I need to reorder the composition visuals such that the selected circle's visual is at the top of the Z-order. I can do so by removing the child visual and adding it to the front of any existing visuals:

```
HR(m_rootVisual->RemoveVisual(circle->Visual.Get()));
HR(m_rootVisual->AddVisual(circle->Visual.Get(), false, nullptr));
```

I also need to keep my list up-to-date, by moving the circle to the front of the list:

```
m_circles.splice(begin(m_circles), m_circles, circle);
```

Paint 5 A Retained-Mode WM_PAINT Message Handler

```
void PaintHandler()
{
    try
    {
        if (IsDeviceCreated())
        {
            HR(m_device3D->GetDeviceRemovedReason());
        }
        else
        {
            CreateDeviceResources();
        }

        VERIFY(ValidateRect(m_window, nullptr));
    }
    catch (ComException const & e)
    {
        ReleaseDeviceResources();
    }
}
```

I then assume the user wants to drag the circle around:

```
SetCapture(m_window);
m_selected = &*circle;
m_mouseX = PhysicalToLogical(point.x, m_dpiX);
m_mouseY = PhysicalToLogical(point.y, m_dpiY);
```

Here, I'm careful to calculate the offset of the mouse position relative to the selected circle. In this way the circle doesn't visually "snap" to the center of the mouse pointer as it's dragged, providing seamless movement. Responding to the `WM_MOUSEMOVE` message allows any selected circle to continue this movement so long as a circle is selected:

```
if (!m_selected) return;

m_selected->SetLogicalOffset(
    PhysicalToLogical(GET_X_LPARAM(lparam), m_dpiX) - m_mouseX,
    PhysicalToLogical(GET_Y_LPARAM(lparam), m_dpiY) - m_mouseY,
    m_dpiX,
    m_dpiY);

HR(m_device->Commit());
```

The `Circle` structure's `SetLogicalOffset` method updates the logical position maintained by the circle, as well as the physical position of the composition visual. I'm also careful to use the `GET_X_LPARAM` and `GET_Y_LPARAM` macros to crack the `LPARAM`, rather than the usual `LOWORD` and `HIWORD` macros. While the position reported by the `WM_MOUSEMOVE` message is relative to the top-left corner of the window, this will include negative coordinates if the mouse is captured and the circle is dragged above or to the left of the window. As usual, changes to the visual tree must be committed for them to be realized. Any movement comes to an end in the `WM_LBUTTONDOWN` message handler by releasing the mouse and resetting the `m_selected` pointer:

```
ReleaseCapture();
m_selected = nullptr;
```

Finally, I'll conclude with the best part. The most compelling proof that this is indicative of retained-mode graphics is when you consider the `WM_PAINT` message handler in **Figure 5**.

The `CreateDeviceResources` method creates the device stack up front. As long as nothing goes wrong, no further work is done by the `WM_PAINT` message handler, other than to validate the window. If device loss is detected, the various catch blocks will release the device and invalidate the window as necessary. The next `WM_PAINT` message to arrive will again recreate the device resources. In my next column I'll show you how you can produce visual effects that aren't directly driven by user input. As the composition engine performs more of the rendering without involving the application, it's possible that device loss might occur without the application even knowing it. That's why the `GetDeviceRemovedReason` method is called. If the composition engine detects device loss it will send the application window a `WM_PAINT` message purely so it can check for device loss by calling the `Direct3D` device's `GetDeviceRemovedReason` method. Take `DirectComposition` for a test drive with the accompanying sample project! ■

KENNY KERR is a computer programmer based in Canada, as well as an author for *Pluralsight* and a Microsoft MVP. He blogs at kennykerr.ca and you can follow him on Twitter at twitter.com/kennykerr.

THANKS to the following Microsoft technical experts for reviewing this article: *Leonardo Blanco and James Clarke*

We didn't invent the Internet...

...but our components help you power the apps that bring it to business.



TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- **E-Business**
AS2, EDI/X12, NAESB, OFTP ...
- **Credit Card Processing**
Authorize.Net, TSYS, FDMS ...
- **Shipping & Tracking**
FedEx, UPS, USPS ...
- **Accounting & Banking**
QuickBooks, OFX ...
- **Internet Business**
Amazon, eBay, PayPal ...
- **Internet Protocols**
FTP, SMTP, IMAP, POP, WebDav ...
- **Secure Connectivity**
SSH, SFTP, SSL, Certificates ...
- **Secure Email**
S/MIME, OpenPGP ...
- **Network Management**
SNMP, MIB, LDAP, Monitoring ...
- **Compression & Encryption**
Zip, Gzip, Jar, AES ...



The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to

any application, on any platform, anywhere, and you do the rest. Since 1994, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

connectivity
powered by 

To learn more please visit our website →

www.nsoftware.com

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ORLANDO
ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO

**NOV
17-21**



"HELP US VISUAL STUDIO LIVE! – YOU'RE OUR ONLY HOPE!"

THE CODE IS STRONG WITH THIS ONE!



While your development challenges may not be as dire as the Rebellion's, we know you're searching for real-world knowledge and education on the Microsoft Platform. And for 21 years and counting, Visual Studio Live! has been the conference more developers trust for independent, practical training.

EVENT PARTERS

Magenic Microsoft

PLATINUM SPONSOR



GOLD SPONSOR



SUPPORTED BY



Visual Studio

Visual Studio
MAGAZINE

PRODUCED BY





TECH EVENTS WITH PERSPECTIVE

FEATURED SPEAKERS:



Andrew Brust



Rockford Lhotka



Miguel Castro



Deborah Kurata



John Papa



Rachel Appel



Brian Randell



Lenni Lobel



Brian Noyes

REGISTER TODAY AND SAVE \$500!

Use promo code ORLAUG4 by August 6



**5 GREAT
CONFERENCES.**
1 GREAT PRICE.

Visual Studio Live! Orlando is part of Live! 360, the ultimate IT and Developer line-up. This means you'll have access to 5 co-located conferences for 1 low price!

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

SQL Server **LIVE!**
TRAINING FOR DBAs AND IT PROS

SharePoint **LIVE!**
TRAINING FOR COLLABORATION

Modern Apps **LIVE!**
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS

TURN THE PAGE FOR
MORE EVENT DETAILS



Check Out the Additional Sessions for Devs, IT Pros, & DBAs at Live!360



SharePoint LIVE!

TRAINING FOR COLLABORATION

SharePoint Live! features 15+ DEVELOPER sessions, including:

- Workshop: Modern Office 365, SharePoint & Cloud Development Ramp-Up - *Andrew Connell*
- Lean-Agile Development with SharePoint - *Bill Ayres*
- How to Improve the SharePoint UI Using Bootstrap 3 - *Ryan McIntyre*
- Building SharePoint Single Page Apps with AngularJS - *Andrew Connell*
- How to Develop and Debug Client-Side Code - *Mark Rackley*
- Workshop: Apps for SharePoint - The Next Level - *Paul Schaefflein*



SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

SQL Server Live! features 15+ DEVELOPER sessions, including:

- SQL Server 2014 In-memory OLTP Deep Dive - *Scott Klein*
- Dealing with Errors in SQL Server Integration Services - *David Dye*
- Secrets of SQL Server: Database WORST Practices - *Pinal Dave*
- Statistics and the Query Optimizer - *Grant Fritch*
- Busy Developer's Guide to NoSQL - *Ted Neward*
- Workshop: SQL Server 2014 for Developers - *Leonard Lobel*



TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

TechMentor features IT Pro and DBA sessions, including:

- Desired State Configuration: An Administrator's Overview - *Don Jones*
- GO DARK: The PowerShell Delegated Administration Resource Kit (AKA JEA) - *Don Jones*
- Top 20 Mistakes in Microsoft Public Key Infrastructure (PKI) Deployments - *Mark B. Cooper*
- Integrating Office 365 with Active Directory, Step-by-Step - *John O'Neill, Sr*
- Document Everything with Windows PowerShell - *Jeffery Hicks*
- Proactive Security in Windows Environments - *Sami Laiho*

FEATURED SPEAKERS:



Andrew Connell



Matt McDermott



Don Jones



Greg Shields

See the FULL AGENDA at
live360events.com



Scan the QR
code to register or for more event details.

Register at
vslive.com/orlando
Use Promo Code ORLAUG4

CONNECT WITH
VISUAL STUDIO LIVE!



twitter.com/vslive – @VSLive



facebook.com – Search "VSLive"



linkedin.com – Join the "visual studio live" group!

AGENDAS AT-A-GLANCE Visual Studio Live! & Modern Apps Live!

Visual Studio / .NET Framework	JavaScript / HTML5 Client	ASP.NET	Cross-Platform Mobile Development	Cloud Computing	Windows Client	Windows Phone	Data Management	ModernAppsLIVE!
--------------------------------	---------------------------	---------	-----------------------------------	-----------------	----------------	---------------	-----------------	-----------------

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Pre-Conference: Sunday, November 16, 2014						
4:00 PM	9:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center						
6:00 PM	9:00 PM	Dine-A-Round Dinner @ Universal CityWalk						

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Pre-Conference Workshops: Monday, November 17, 2014 (Separate entry fee required)			
6:30 AM	8:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries			
8:00 AM	5:00 PM	VSM01 - Workshop: Native Mobile App Development for iOS, Android, and Windows using C# - Marcel de Vries & Rockford Lhotka	VSM02 - Workshop: AngularJS in 0 to 60 - John Papa	VSM03 - Workshop: From Code to Release: DevOps for Developers - Brian Randall	MAM01 - Workshop: Modern App Technology Overview - Android, iOS, Cloud, and Mobile Web - Nick Landry, Kevin Ford, & Steve Hughes
5:00 PM	6:00 PM	EXPO Preview			
6:00 PM	7:00 PM	Live! 360 Keynote:			

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Day 1: Tuesday, November 18, 2014					
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries					
8:00 AM	9:00 AM	Visual Studio Live! & Modern Apps Live! Keynote					
9:00 AM	9:30 AM	Networking Break • Visit the EXPO					
9:30 AM	10:45 AM	VST01 - Getting Started with Xamarin - Walt Ritscher	VST02 - Great User Experiences with CSS 3 - Robert Boedigheimer	VST03 - What's New in MVC 5 - Miguel Castro	VST04 - New IDE and Editor Features in Visual Studio 2013 - Deborah Kurata	MAT01 - Defining Modern App Development - Rockford Lhotka	
11:00 AM	12:15 PM	VST05 - Building Your First Universal Application for Windows Phone and Windows Store - Brian Peek	VST06 - HTML5 and Modernizr for Better Web Sites - Robert Boedigheimer	VST07 - What's New in Web API 2 - Miguel Castro	VST08 - Visual Studio Data Tools for Developers - Deborah Kurata	MAT02 - Modern App Architecture - Rockford Lhotka	
12:15 PM	2:00 PM	Lunch • Visit the EXPO					
2:00 PM	3:15 PM	VST09 - Introduction to Developing Mobile Apps for the Web Developer - Ben Hoelting	VST10 - Build an Angular and Bootstrap Web Application in Visual Studio from the Ground Up - Deborah Kurata	VST11 - ASP.NET MVC for Mobile Devices - Rob Daigneau	VST12 - The Road to Continuous Delivery, Automated UI Testing for Web, WPF and Windows Forms - Marcel de Vries	MAT03 - ALM with Visual Studio Online (TFS) and Git - Brian Randell	
3:15 PM	4:15 PM	Networking Break • Visit the EXPO					
4:15 PM	5:30 PM	VST13 - Creating Responsive Cross-Platform Native/Web Apps with JavaScript and Bootstrap - Ben Dewey	VST14 - Hate JavaScript? Try TypeScript - Ben Hoelting	VST15 - What's New in WPF 4.5 - Walt Ritscher	VST16 - Understanding Dependency Injection & Writing Testable Software - Miguel Castro	MAT04 - Reusing Business and Data Access Logic Across Platforms - Kevin Ford	
5:30 PM	7:30 PM	Exhibitor Reception					

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Day 2: Wednesday, November 19, 2014					
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries					
8:00 AM	9:00 AM	Live! 360 Keynote:					
9:15 AM	10:30 AM	VSW01 - Build Cross-Platform Apps with Shared Projects, CSLA .NET, and C# - Rockford Lhotka	VSW02 - AngularJS Jump-Start - John Papa	VSW03 - Best Practices for Self Hosting Web API Based Services - Rob Daigneau	VSW04 - The Busy Developers Guide to Virtualization with Hyper-V - Brian Randall	MAW01 - Coding for Quality and Maintainability - Jason Bock	
10:30 AM	11:00 AM	Networking Break • Visit the EXPO					
11:00 AM	12:15 PM	VSW05 - Building Cordova Apps in Visual Studio 2013 - Nick Landry	VSW06 - AngularJS Anywhere with Node.js - John Papa	VSW07 - Slice Development Time With ASP.NET MVC, Visual Studio, and Razor - Philip Japikse	VSW08 - Build It and Ship It with TFS and Release Management - Brian Randall	MAW02 - UX Design for Modern Apps - Anthony Handley	
12:15 PM	1:45 PM	Birds-of-a-Feather Lunch • Visit the EXPO					
1:45 PM	3:00 PM	VSW09 - What's New in Azure for Developers - Vishwas Lele	VSW10 - I Just Met You, and "This" is Crazy, But Here's My NaN, So Call(me) Maybe? - Rachel Appel	VSW11 - Automated Cross Browser Testing of Your Web Applications with Visual Studio CodedUI - Marcel de Vries	VSW12 - Readable Code - John Papa	MAW03 - Applied UX: iOS, Android, Windows - Anthony Handley	
3:00 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m.					
4:00 PM	5:15 PM	VSW13 - API Management - Vishwas Lele	VSW14 - Creating HTML5 and Angular Websites Using Visual Studio LightSwitch - Michael Washington	VSW15 - Build Real-Time Websites and Apps with SignalR - Rachel Appel	VSW16 - Visual Studio Online: An Update Every Three Weeks - Brian Randall	MAW04 - Leveraging Azure Services - Kevin Ford	
8:00 PM	10:00 PM	Live! 360 Evening Event					

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Day 3: Thursday, November 20, 2014					
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries					
8:00 AM	9:15 AM	VSH01 - Creating Visual Studio Cloud Business Applications for Office 365 / SharePoint 2013 - <i>Michael Washington</i>	VSH02 - To Be Announced	VSH03 - Games Development with Unity and Other Frameworks for Windows and Windows Phone - <i>Brian Peek</i>	VSH04 - Managing the .NET Compiler - <i>Jason Back</i>	MAH01 - Analyzing Results with Power BI - <i>Steve Hughes</i>	
9:30 AM	10:45 AM	VSH05 - Microsoft Azure Web Sites for the Web Developer - <i>Eric D. Boyd</i>	VSH06 - Building Rich Data Input Windows 8 Applications - <i>Brian Noyes</i>	VSH07 - Build Your First Mobile App in 1 Hour with Microsoft App Studio - <i>Nick Landry</i>	VSH08 - Writing Asynchronous Code Using .NET 4.5 and C# 5.0 - <i>Brian Peek</i>	MAH02 - Building a Native iOS App - <i>Lou Miranda</i>	
11:00 AM	12:15 PM	VSH09 - Moving Web Apps to the Cloud - <i>Eric D. Boyd</i>	VSH10 - XAML Antipatterns - <i>Ben Dewey</i>	VSH11 - To Be Announced	VSH12 - Asynchronous Debugging in .NET - <i>Jason Back</i>	MAH03 - Building an Android App with Xamarin - <i>Nick Landry</i>	
12:15 PM	1:30 PM	Lunch on the Lanai					
1:30 PM	2:45 PM	VSH13 - To Be Announced	VSH14 - Building Maintainable and Extensible MVVM WPF Apps with Prism 5 - <i>Brian Noyes</i>	VSH15 - Learning Entity Framework 6 - <i>Leonard Lobel</i>	VSH16 - Rock Your Code Using Code Contracts - <i>David McCarter</i>	MAH04 - Building a Windows App - <i>Brent Edwards</i>	
3:00 PM	4:15 PM	VSH17 - To Be Announced	VSH18 - XAML for the WinForms Developer - <i>Philip Japikse</i>	VSH19 - Database Development with SQL Server Data Tools - <i>Leonard Lobel</i>	VSH20 - Rock Your Code With Visual Studio Add-ins - <i>David McCarter</i>	MAH05 - Building a Responsive Single Page App - <i>Allen Conway</i>	
4:30 PM	5:45 PM	Live! 360 Conference Wrap-up					

START TIME	END TIME	Visual Studio Live! & Modern Apps Live! Post-Conference Workshops: Friday, November 21, 2014 (Separate entry fee required)						
7:30 AM	8:00 AM	Post-Conference Workshop Registration - Coffee and Morning Pastries						
8:00 AM	5:00 PM	VSF01 - Workshop: Service Orientation Technologies: Designing, Developing, & Implementing WCF and the Web API - Miguel Castro		VSF02 - Workshop: Build a Windows 8.1 Application in a Day - Philip Japikse			MAF01 - Workshop: Modern App Development In-Depth: iOS, Android, Windows, and Web - Brent Edwards, Anthony Handley, Lou Miranda, & Allen Conway	

Speakers and sessions subject to change



Cool (and Free) Tools for Entity Framework

Now that Entity Framework is open source, the development community is able to contribute code to it at entityframework.codeplex.com. But don't limit yourself to looking there for tools and extensions. There are other great tools, both commercial and community-driven, that can help you achieve more with Entity Framework. In this column, I want to highlight some of those tools and extensions that come from the community. They're all free and available through the Visual Studio Gallery (for IDE extensions) or NuGet (for code libraries). There are even some from members of the Entity Framework team that are pet projects and external to the official bits.

EntityFramework Reverse POCO Code First Generator

There are still a lot of folks who think the Database First workflow is the only way to create a model for Entity Framework from an existing database. That particular workflow creates an EDMX file, which allows you to interact with your model in a designer. Using the designer also means you're dependent on a code generator to create your entity classes. But it's possible to use EF Code First with a pre-existing database, thanks to reverse-engineering tools. There are now three ways to accomplish this. My favorite is this template: EntityFramework Reverse POCO Code First Generator by Simon Hughes (efreversepoco.codeplex.com). To be fair, the other two are Microsoft-created tools I've also used extensively. One is a feature of the EF Power Tools Beta, which you can see in action in my May 2011 blog post at bit.ly/1ky2SAJ. The other is an enhancement in the Visual Studio EF Designer that was in the release of Entity Framework 6.1 Tools for Visual Studio. The EF Designer now includes a visual way to create POCOs from an existing database, and solves a problem that agonized me for a long time by providing the ability to select which tables to include in a model. I demonstrated this feature in my February 2014 blog post at bit.ly/1r0qUv4.

Hughes' template was first released onto CodePlex in October 2012, and it had the ability to select subsets of a database from the start. In fact, the tool seems to provide all of the features I've hoped would be added or improved in the EF Power Tools and in the newer EF6.1 Designer. And I'm not the only one to feel this way: As I'm writing this column, there have been nearly 77,000 downloads of the EntityFramework Reverse POCO Code First Generator.

You can install the template through the Visual Studio Extensions and Updates feature and have offline access to it for future projects, or you can use it directly from the Web whenever you want to generate POCOs from a database. Before adding the template, you should add Entity Framework to your project and explicitly specify the database connection string in your config file, because the template relies on these. Once you have EF and your connection string, you initiate the template by adding a new item to a project in Visual Studio. If you've installed the template, you'll find it under Installed; otherwise, open the Online section. Notice I haven't filtered the templates in **Figure 1**, but the EntityFramework Reverse POCO Code First Generator is at the top of the list sorted by Most Popular items. I was certainly impressed by that position.

Selecting this item results in a new `database.tt` template file in your project. The template has a slew of configurations with default settings already applied. You can see a slice of the template file in **Figure 2**, which shows some of the simpler configurations you can use to impact the generated code. The `ElementsToGenerate` setting lets you control what types of classes are generated (POCOs, context, fluent configurations, `UnitOfWork`). One benefit of this configuration is that you can generate different items for different projects. You can control namespaces and how types are named. You can generate some basic Windows Communication Foundation (WCF) classes, and more. The configurations also let you specify what database tables should be included/excluded by using RegExes to specify the patterns of table or view names. Specifying objects has been an important feature for me. However, now that the EF6.1 Designer supports visually selecting database objects for creating a Code First model, I find that specifying the object names in the `.tt` file a bit clunky in comparison.

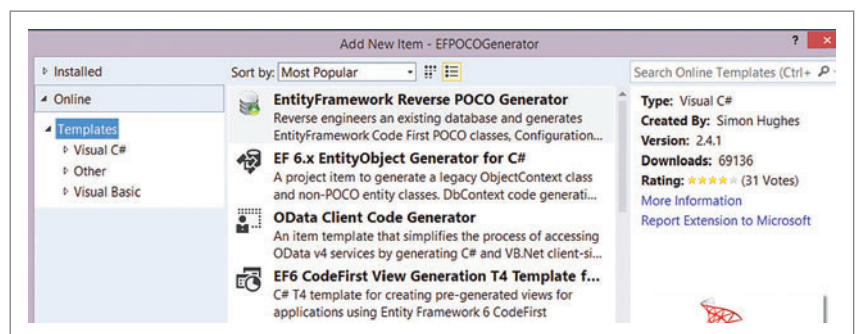


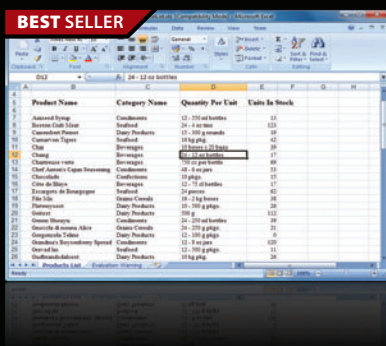
Figure 1 Getting the EntityFramework Reverse POCO Code First Generator



Aspose.Total for .NET from \$2,449.02

Every Aspose .NET component in one package.

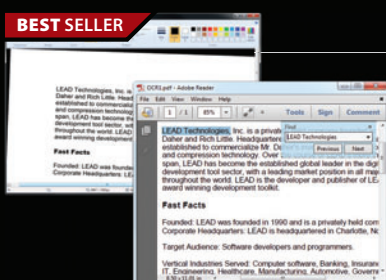
- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Work with charts, diagrams, images, project plans, emails, barcodes, OCR and OneNote files alongside many more document management features in .NET applications
- Common uses also include mail merging, adding barcodes to documents, building dynamic reports on the fly and extracting text from PDF files



LEADTOOLS Document Imaging SDKs V18 from \$2,695.50

Add powerful document imaging functionality to desktop, tablet, mobile & web applications.

- Comprehensive document image cleanup, preprocessing, viewer controls and annotations
- Fast and accurate OCR, OMR, ICR and Forms Recognition with multi-threading support
- PDF & PDF/A Read / Write / Extract / View / Edit
- Barcode Detect / Read / Write for UPC, EAN, Code 128, Data Matrix, QR Code, PDF417
- Zero-Footprint HTML5/JavaScript Controls & Native WinRT Libraries for Windows Store



ComponentOne Studio Enterprise 2014 v2 from \$1,315.60

.NET Tools for the Professional Developer: Windows, HTML5/Web, and XAML.

- Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers
- Includes 40+ UI widgets built with HTML5, jQuery, CSS3 and SVG
- New Sparkline control for HTML5 and Web Forms and options for exporting Data views
- New tools to connect, manage and display data in modern Web & Windows interfaces
- All Microsoft platforms supported, Visual Studio 2013, ASP.NET, WinForms, WPF & more



Help & Manual Professional from \$583.10

Easily create documentation for Windows, the Web and iPad.

- Powerful features in an easy accessible and intuitive user interface
- As easy to use as a word processor, but with all the power of a true WYSIWYG XML editor
- Single source, multi-channel publishing with conditional and customized output features
- Output to HTML, WebHelp, CHM, PDF, ePub, RTF, e-book or print
- Styles and Templates give you full design control



Another significant advantage the Reverse POCO Code First Generator had over the reverse-engineer feature of the EF Power Tools was performance. However, the new EF6.1 Designer speeds up the code generation dramatically, so it's no longer a determining factor for me. Nevertheless, the Reverse POCO Code First Generator has so many nice features (such as including default values for columns as default values for properties) and ways to customize how the classes are generated (such as specifying what type of collection should be used for navigation properties) that I still like it. I just wish I could merge it with the EF Designer.

There's so much in this template that, rather than listing all of its features here, I'll point you to the documentation and 30-minute demonstration video (reversepoco.com) so you can get an excellent head start on using this template. You'll also see that Hughes has been accepting contributions to the project on CodePlex. An important note to keep in mind is that the EntityFramework Reverse POCO Code First Generator currently works with SQL Server and SQL Server CE. The EF6.1 Designer works with these databases and any other database providers that have been updated to work with EF6.1. I've been told that the next major version (v3) of Hughes' generator will be able to reverse engineer other databases, such as Oracle, MySQL and so on, in the same way that EF6.1 Designer can.

Entity Framework Utilities

The next library I'm highlighting is maintained by Mikael Eliasson on GitHub (bit.ly/UeWHGW). You can get the utilities into your projects by searching for EFUtilities in NuGet.

This set of utilities focuses on performance and adds functionality for something long missed by Entity Framework developers: the ability to perform bulk operations—insert, update and delete—in the database. The EFBulkOperation class in these utilities works with DbContext, so you can't use it with theObjectContext. Although a number of developers have attacked the problem, I've known Mikael for a while so I naturally gravitated to his offering. There's also discussion on the CodePlex site for Entity Framework about best approaches to getting bulk operations into the native API.

The utility allows you to write LINQ queries on the EFBulkOperation class in order to execute the relevant operation. As a test, I created a model (using the EntityFramework Reverse POCO Code First Generator tool I just discussed) from the AdventureWorks2012 database. Then, in my code, I created a list of 1,000 instances of the BusinessEntity type:

```
var entityList = new List<BusinessEntity>();
for (int i = 0; i < 1000; i++)
{
    entityList.Add(new BusinessEntity());
}
```

After installation, the database.tt template file has a slew of configurations you can set.

I inserted these using standard Entity Framework code and then called SaveChanges:

```
using (var context = new AWModel()) {
    context.BusinessEntities.AddRange(entityList);
    context.SaveChanges();
}
```

This caused 1,000 individual insert statements to be sent to my SQL Server database, and the complete operation took an average of about 6 seconds over a number of repeated executions. I have removed EF warm-up times from this timing.

Then I used the EFBatchOperation class to insert the same 1,000 BusinessEntity objects:

```
using (var context2 = new AWModel()) {
    EFBatchOperation.For(context2, context2.BusinessEntities).InsertAll(entityList);
}
```

The average execution was about 12 milliseconds.

Keep in mind that in the standard EF code, I had to add all of the instances to the context so that the change tracker could determine what SQL needed to be sent to the server. The EFBatchOperation class, on the other hand, doesn't rely on the EF change tracker to build the

SQL. Instead, it creates a DataReader that's a class in the EFUtilities, and streams that to the database using the .NET SqlBulkCopy class that was introduced way back in ADO.NET 2.0. I was happy to see this because it has always been the path I've recommended to solve this problem. So not only is this method not forcing the context to track 1,000 entities, it's also sending a single command to the database along with the binary stream of the data to be inserted. SQL Profiler won't display the stream, so all I'll see in SQL Profiler as a result of the EFBatchoperation.InsertAll method is this single command:

```
insert bulk BusinessEntity
([rowguid] UniqueIdentifier,
[ModifiedDate] DateTime)
```

```
// Misc settings *****
// Namespace = ""; // Override the default namespace here
DbContextName = "MyDbContext";
ConnectionStringName = "MyDbContext"; // Searches for this connection string in config files listed below
ConfigurationClassName = "Configuration"; // Configuration, Mapping, Map, etc. This is appended to the Poco
ConfigFilenameSearchOrder = new[] { "app.config", "web.config", "app.config.transform",
                                     "web.config.transform" }; // Add more here if required. The config file:

MakeClassesPartial = false;
GenerateSeparateFiles = true;
UseCamelCase = true; // This will rename the tables & fields to use CamelCase. If false table & field names
IncludeComments = true; // Adds comments the generated code
IncludeViews = true;
DisableGeographyTypes = false; // Turns off use of System.Data.Entity.Spatial.DbGeography and System.Data.Entity
CollectionType = "List"; // Determines the type of collection for the Navigation Properties. "ObservableCollection"
CollectionTypeNamespace = ""; // "System.Collections.ObjectModel" is required if setting the CollectionType

// Elements to generate *****
// Add the elements that should be generated when the template is executed.
// Multiple projects can now be used that separate the different concerns.
ElementsToGenerate = Elements.Poco | Elements.Context | Elements.UnitOfWork | Elements.PocoConfiguration;

// Use these namespaces to specify where the different elements of the template are being generated.
PocoNamespace = "";
ContextNamespace = "";
UnitOfWorkNamespace = "";
PocoConfigurationNamespace = "";
```

Figure 2 A Section of the EntityFramework Reverse POCO Generator Configuration File



File APIs for Developers: DOC, XLS, PPT, PDF, EML and more

A Visual Studio 2013 Q&A with
Justin Anderson of Aspose Business Team

Q. Who is Aspose, and why should developers care?

A. Aspose makes developer APIs for working with business-centric files, like: .DOC, .XLS, .PPT, .PDF, .EML and many more. Visual Studio 2013 does not offer native support for working with business files, and no other company offers the comprehensive support that Aspose does.

Q. What types of things can be done with files?

A. Developers use files in a number of ways: importing and exporting data, reporting, user collaboration and data visualization, just to name a few. Aspose APIs allow developers to: create files, convert files from one format to another, combine files, edit file content and formatting, and print files.

Q. What makes Aspose special?

A. Quality and comprehensiveness; no one else supports files like Aspose. Whether you look at the number of supported file formats, file features or platforms, Aspose is the leader. We offer native APIs for almost every major platform.

Q. Does Aspose use automation, or have third-party dependencies?

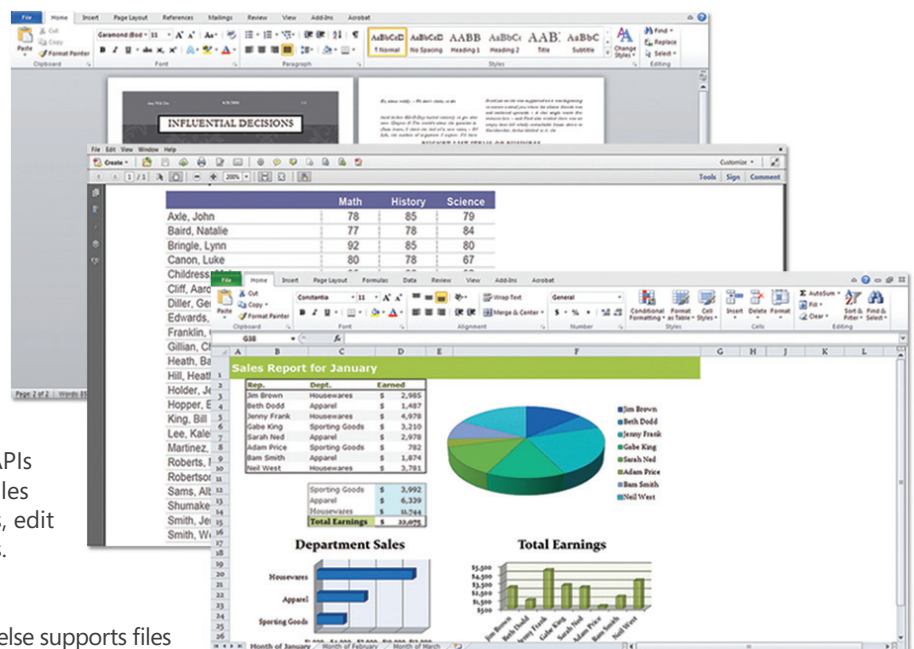
A. No. Aspose APIs are 100% standalone. They are a pure .NET alternative to the Microsoft Office Object Model.

Q. Who uses Aspose?

A. A lot of folks; our customers range from small shops, to the largest enterprise companies. Aspose has over 14,000 customers in 104 countries, and we serve more than 70% of the Fortune 100.

Q. Is it hard or costly to get started with Aspose?

A. Not at all; Aspose offers one of the best "try before



you buy" models around. Free evaluation versions are available for all products. Aspose also provides free support, even before a purchase is made. Evaluation licenses are available in case developers need to showcase how full versions work.

Aspose believes in the quality of their products, and they want developers to ensure their APIs are a perfect fit, risk free.

Q. What about the cloud; does Aspose have a cloud solution?

A. Absolutely! Aspose products work just fine in the cloud. Aspose also offers a completely hosted solution, which is pay as you go. Developers can setup a free account and begin testing within minutes.

For more information, please visit →

www.aspose.com

A nice protection here is that if the database provider you're using doesn't have a bulk method, the utility will revert to the EF default way of handling the request. In this case, that would mean sending 1,000 insert commands to the database.

EFBatchOperation also has methods to do bulk updates and bulk deletes, and what's nice about each of these is that you don't have to load the entities into memory or the context in advance as you would with EF. Instead, here's an example of how you'd delete a selection of rows from the database using EFBatchOperation:

```
EFBatchOperation.For(context, context.BusinessEntityContacts).Where(
    p => p.ContactTypeId == 11).Delete();
```

It sends the relevant SQL to the database:

```
DELETE FROM [Person].[BusinessEntityContact] WHERE 11 = [ContactTypeID]
```

The bulk operation commands already provide significant performance gains, and now Eliasson is working on a replacement for the DbContext.Include method to improve the performance of eager loading. Rather than flatten the data and waste resources on a lot of redundant data in the results, the goal of the method in EF Utilities is to "load every Included collection as a separate query instead and then manually fix the relations. The goal is also to provide child collection filtering, sorting and, if possible, paging." You can read more about this feature and keep an eye on its progress at bit.ly/1hBsGf5. Don't overlook the readme.md file for this project.

Entity Framework 6 Contrib

Entity Framework 6 Contrib is a project on CodePlex by Microsoft MVP Unai Zorrilla (ef6contrib.codeplex.com). Zorrilla has made a number of contributions to the Microsoft EF package, such as Custom Pluralization, AddRange and RemoveRange, and AddFromAssembly—all features I've written about in past columns or articles. Zorrilla has a handful of additional helpers he maintains on the ef6contrib.codeplex.com site, including:

- A preconfigured Spanish pluralization service (which has been incorporated into the EntityFramework Reverse POCO Code First Generator project I discussed earlier).

Figure 3 Configuring Some Analyzers from Entity Framework 6 Contrib

```
public class Configuration : DbConfiguration {
    public Configuration() {
        this.AddDependencyResolver(new CustomResolver());

        this.AddInterceptor(new PerformanceInterceptor((msg => {
            Console.WriteLine(msg.Message);
        })));
    }

    private class CustomResolver : IDbDependencyResolver {
        public object GetService(Type type, object key) {
            return null;
        }

        public IEnumerable<object> GetServices(Type type, object key) {
            // You can use here your preferred IoC container
            if (typeof(IPerformanceAnalyzer).IsAssignableFrom(type)) {
                return new List<IPerformanceAnalyzer>() {
                    new ExecutionTimePerformanceAnalyzer(TimeSpan.FromSeconds(3)),
                    new UnparametrizedWhereClausesPerformanceAnalyzer(),
                    new TopSlowQueriesPerformanceAnalyzer(10)
                };
            }
            return Enumerable.Empty<object>();
        }
    }
}
```

- Two more-convenient methods for adding Fluent API configurations to a Code First model at run time.
- A number of operations you can use with Code First migrations that aren't included in EF, such as DatabaseCollation, GrantTablePermission and RevokeTablePermission.
- An initializer that works with migrations called DropAndMigrateDatabaseToLatestVersion.
- A set of query analyzer types that leverage EF6 interceptors and dependency resolvers.

What I've always loved about Zorrilla's additions to EF is that he takes code he's written and used repeatedly in his own work and encapsulates it to share with others. The analyzers, in particular, are extra cool. You can set up the parameters by which you want to measure performance of the EF database execution and monitor how well your code aligns with your expectations. There's also an analyzer that checks for non-parameterized queries. If you add the interceptor to the DbConfiguration class (another EF6 feature), it will analyze all of the commands from a given DbContext. The CodePlex project has a bunch of tests and samples that let you explore the analyzers, as well as the other tools included in Entity Framework 6 Contrib. **Figure 3** shows a DbConfiguration class from the samples that adds the PerformanceInterceptor and then specifies it should check three of the four analyzers in the collection.

Based on the configuration, the console window will display any reports from the analyzers.

EF Extensions and Tools Galore

These are just a few of the tools I've found to be really beneficial to my work. There are others I want to make sure you don't overlook, but I don't have room to dig into all of them here. Erik Ejlskov Jensen has a list of some of the more notable extensions in his blog post: "Entity Framework 6 (& SQL Server Compact) (5)–Entity Framework 6 Extensions" (bit.ly/SxPc5). The list includes an updated version of the EF Caching Provider, which enables second-level caching for EF6 (bit.ly/UePlmU); Trackable Entities, a replacement for the discontinued Self-Tracking Entities (trackable.codeplex.com); and some additional projects that resolve bulk operations functionality. Ejlskov Jensen is a SQL Server CE MVP and in addition to contributing lots of great SQL CE-related functionality directly to EF6, he has an extremely popular Visual Studio extension, SQL Server Compact Toolbox, that includes many features to help with EF. If you use SQL CE with or without EF, the toolbox is an essential extension. Finally, I want to add a shout-out about work that Jimmy Bogard has done in a series of blog posts he calls "Missing EF Feature Workarounds," including one that extends EF6 to provide global filters to an EF Model and is available as a NuGet package. You can learn more about this in his blog post at bit.ly/1prZv0a. ■

JULIE LERMAN is a Microsoft MVP, .NET Framework mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework" (2010) as well as a Code First edition (2011) and a DbContext edition (2012), all from O'Reilly Media. Follow her on Twitter at twitter.com/julielerman and see her Pluralsight courses at julieme/PS-Videos.

THANKS to the following technical experts for reviewing this article: Mikael Eliasson, Simon Hughes and Unai Zorrilla

Spreadsheets Made Easy.



Fastest Calculations

Evaluate complex Excel-based models and business rules with the fastest and most complete Excel-compatible calculation engine available.



Comprehensive Charting

Enable users to visualize data with comprehensive Excel-compatible charting which makes creating, modifying, rendering and interacting with complex charts easier than ever before.



Windows
Forms



Silverlight



WPF

Powerful Controls

Add powerful Excel-compatible viewing, editing, formatting, calculating, filtering, sorting, charting, printing and more to your WinForms, WPF and Silverlight applications.



Scalable Reporting

Easily create richly formatted Excel reports without Excel from any ASP.NET, Windows Forms, WPF or Silverlight application.

Download your free fully functional evaluation at SpreadsheetGear.com



SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com

Integrating OneDrive into Your Windows Store Apps

Tony Champion

You can't go very far today without running into a conversation about cloud development. Public, private, on- or off-premises, cloud offerings now come in all shapes and sizes. But when talking about consumer-based apps that deal with user files, chances are you'll focus on offerings like Microsoft OneDrive.

Microsoft OneDrive, recently rebranded, gives consumers a convenient way to store, share and access their personal files from anywhere on any device. OneDrive is now fully integrated into all Microsoft platforms, and it's compatible with most other platforms currently in use. Documents, photos, videos, and many other types of files can be easily shared, synced or accessed across devices. So, if you're building an app that involves user files, integrating OneDrive into your app should be an important feature.

If you're building Windows Store apps, you automatically get a certain level of OneDrive capability for free. In Windows 8.1, you'll find a Windows Store OneDrive app, as well as desktop integration in the file system that includes an implementation of the File Open Picker and File Save Picker contracts. This means if you use the file open or save pickers in your app, the user will automatically have the ability to open or save documents to his OneDrive account. The same is true for the open and save file dialogs in desktop apps.

This article discusses:

- Using the Live SDK to access the contents of OneDrive
- What you need to do to integrate OneDrive into your app
- Objects on OneDrive
- Working with files and folders on OneDrive

Technologies discussed:

OneDrive API, Visual Studio 2013 Update 2, Live SDK version 5.6, Windows 8.1 Store, XAML

Code download available at:

msdn.microsoft.com/magazine/msdnmag0814

However, what happens if you want a little more control over how this interaction works? What if you want to display the user's data in a different format, or handle the upload or download of multiple files in the background? To address such scenarios, Microsoft provides an API that can be accessed from any platform to allow developers to integrate OneDrive into their solutions. This article explores that API and looks at what's involved in using this API in your Windows Store apps.

Introducing the Live SDK

Access to the contents of OneDrive is facilitated by the Live SDK. At its core, the Live SDK is a collection of JSON-based REST APIs that can be consumed on any platform. The Live SDK uses single sign-on for Windows Store and Windows Phone Store apps and OAuth 2.0 authentication for other platforms. To help developers create successful applications using OneDrive, Microsoft has also released client-specific SDKs. There are currently SDKs for Microsoft Windows and Windows Phone platforms, as well as SDKs for Android and iOS. Links to all of these SDKs can be found at msdn.microsoft.com/onedrive/dn630256. The Live SDK can also be added via NuGet (nuget.org/packages/LiveSDK).

With regard to OneDrive, the primary role of the Live SDK is to allow programmatic access to the files and folders within a user's account. Here are the basic services provided by the SDK:

- Navigate folder hierarchy
- Create and delete new folders
- Read and modify folder properties
- Upload and download files
- Read and modify file properties
- Copy and move files and folders
- Get links to share for files and folders
- Set permissions on files and folders

I use Visual Studio 2013 Update 2 and the Live SDK version 5.6 to explore how to use the SDK in a Windows 8.1 Store app written in XAML. With the availability of client-specific SDKs for most

platforms, the same concepts can easily be migrated to the language and platform of your choice.

Developing Apps with the Live SDK

Before you can start developing apps using the Live SDK, there are a few housekeeping items to address. Because the Live SDK isn't part of the core Windows SDK, you must download the Live SDK and then add a reference to it from your app. To add the reference, just right-click your solution in Visual Studio 2013 and select Add Reference. The Live SDK can be found in the Windows 8.1 Extensions section of the Reference Manager, as shown in **Figure 1**.

By default, a Windows 8.1 Store app has the Internet (Client) capability enabled in its appxmanifest. This is a required capability for using the Live SDK and your project must have it enabled. To verify or enable this capability, open your project's appxmanifest in the designer and make sure Internet (Client) is checked in the Capabilities tab.

Once the Live SDK has been added to your project, you'll find the entire SDK in the Microsoft.Live namespace. However, using the SDK at this point will produce a null object reference error. This is because you must register your app with Live services before you

can access OneDrive. Depending on the platform on which you're developing, this can be done in several ways. For a Windows Store app, you simply associate your app with the Windows Store. The most direct way to do this is to use the wizard in Visual Studio.

To launch the wizard, right-click the Store project in the Solutions Explorer and select Store | Associate App with the Store. The wizard will open with a description of what you'll need in order to continue. Selecting Next prompts you to sign in to a Microsoft account associated with the Windows Store (if you're not already signed in). You might also have to go through a second verification process, such as receiving an authorization code via a text message to the phone number associated with your account. Once you've completed the sign-in process, you'll see a list of all reserved app names associated with your account, as shown in **Figure 2**. If this is a new app, you can also reserve a new app name from this screen. Once you have a name to associate your app with, selecting Next will take you to a summary screen and clicking Associate will complete the process.

Your app is now set up and ready to begin using the Live SDK.

Accessing the User's Data

The first step to enabling OneDrive integration into your app is gaining the ability to access the user's information. As previously mentioned, the Live SDK uses OAuth 2.0 for authorization. However, there are a few additional points to consider aside from simply authenticating your app.

Proper Use of OneDrive If you're familiar with developing and deploying Windows Store apps, you know there are quite a few guidelines detailing what you can and can't do within your app. OneDrive adds its own set of guidelines that must also be taken into consideration.

The main reason for the restrictions is to keep you from undermining the user's trust in OneDrive. When a user has a central source for storing her personal information, she has an inherent trust in that source or she wouldn't be using it. By accessing OneDrive with your app, the user is extending that trust to your app. So, if your app begins deleting the user's data without her consent or knowledge, you undermine that trust and affect not only the user's perception of your app, but also her perception of OneDrive. For that reason, you should take extreme care in how your app interacts with OneDrive, especially when performing functions that update or remove user information. A description of these guidelines can be found at bit.ly/1rQ8iXK.

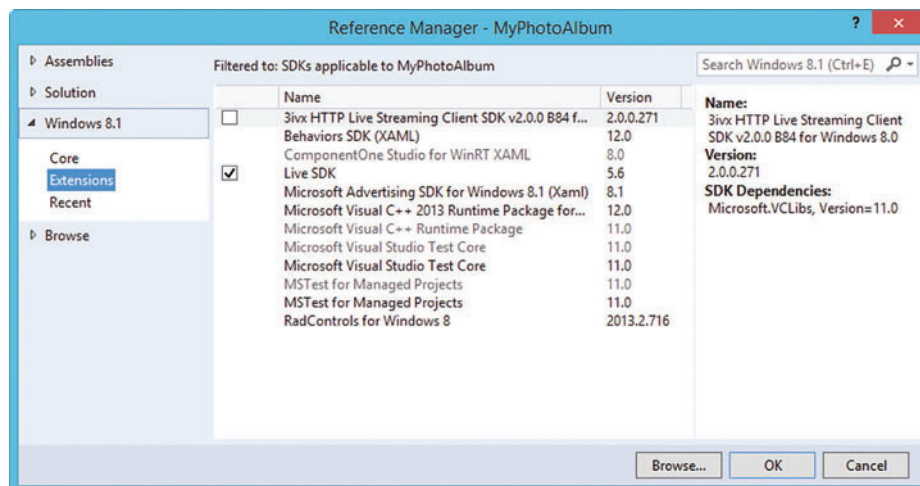


Figure 1 Adding a Reference to Live SDK

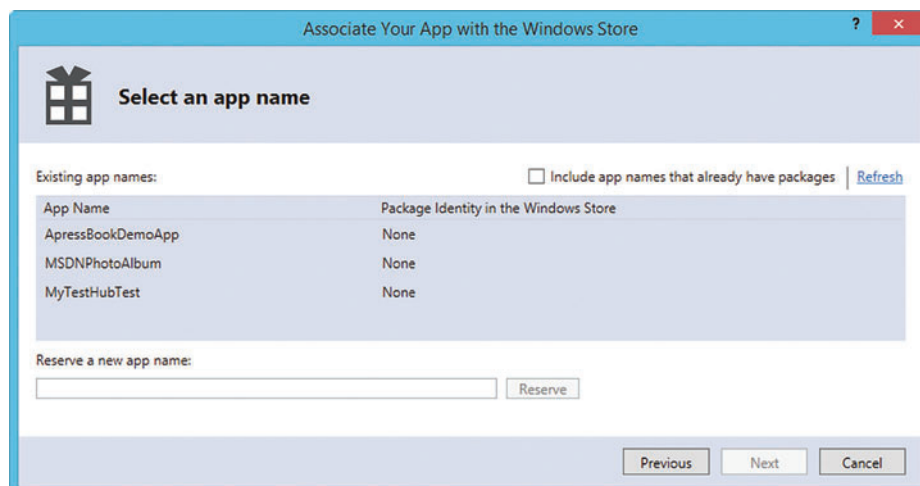


Figure 2 Associating a Reserved Name with an App

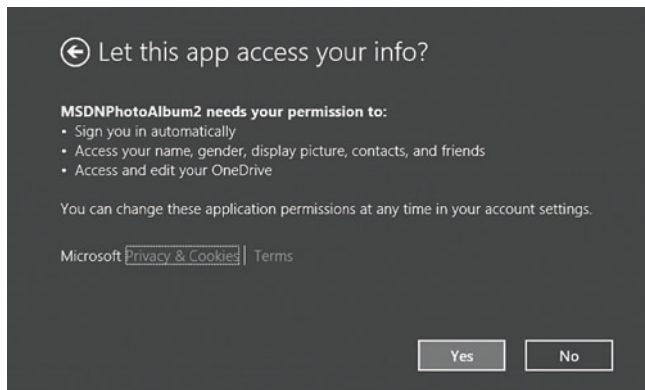


Figure 3 Requesting User Consent to Access OneDrive

Using Single Sign-On When using the Live SDK in a Windows Store app, Microsoft suggests you take advantage of its single sign-on (SSO) features. Just as with the Live SDK, adding SSO to your app involves a few additional requirements, like a privacy statement and an account settings panel. Full details on the steps to add the necessary components to your app can be found at bit.ly/TJWtXB.

After your app is prepared for SSO, logging into the user's account is done using an instance of the `LiveAuthClient` class. `LiveAuthClient` has a `LoginAsync` method that will use the credentials of the Microsoft account that's currently signed in to the Windows 8.1 device. If sign-in is successful, the returned `LiveLoginResult` object will contain a `LiveConnectSession` instance that will be used for all calls to the Live SDK. The following will log the user in and return the Session object:

```
LiveAuthClient authClient = new LiveAuthClient();
LiveLoginResult authResult = await authClient.LoginAsync(new
List<string>() { "wl.signin",
    "wl.basic", "wl.skydrive", "wl.skydrive_update" });
if (authResult.Status == LiveConnectSessionStatus.Connected)
{
    // Add code to handle session held in the Session property
}
```

Scopes You probably noticed the use of scopes like "wl.signin" in the previous code. When your app logs in to a user's account, it must identify what type of access it needs. If you attempt to use portions of the API you haven't requested access for, your attempts will fail. For Windows Store apps, this is no different than attempting to use portions of the Windows SDK that you haven't requested access for in the capabilities section of the package appxmanifest.

The examples in this article will use only the four scopes. The `wl.signin` scope allows you to take advantage of the SSO behavior you should use in Windows Store apps. The `wl.basic` scope enables access to some common information about the user.

For OneDrive applications, there are only two additional scopes you need to be concerned with: `wl.skydrive` and `wl.skydrive_update`. As the names probably suggest, `wl.skydrive` is necessary for read and browse access to a user's OneDrive account. If your app needs the ability to upload files, create or delete folders, or change an object's properties, you use the `wl.skydrive_update` scope. This scope also grants you read permission to OneDrive, so you don't need to include both `wl.skydrive` and `wl.skydrive_update` scopes in the access list.

In order to maintain backward compatibility with apps using the previous branding, OneDrive scope names haven't been changed.

However, I wouldn't be surprised to see a couple of additional scopes using the OneDrive branding appear in future versions of the API.

Because the Live SDK covers a lot more ground than just OneDrive, there are many other scopes that I won't address in this article. You can find a complete descriptive list at msdn.microsoft.com/library/dn631845.

Obtaining User Consent The first time a user runs an app that requests access to his OneDrive account, he will be prompted to grant that app permission to the scopes provided in the login call. **Figure 3** shows an example of such a prompt in a Windows Store app. Once the user grants permission, he won't be prompted again as long as the scope list doesn't change. If an updated version of the app requests additional features of the API, the user will again be prompted for permission.

Objects on OneDrive

The OneDrive API considers everything an object. Whether folder, photo, or spreadsheet, everything stored in OneDrive has a set of similar properties and they share API calls. Each object has a collection of properties and may have a collection of child objects. The

Figure 4 JSON Returned from a Folder Object Query

```
{
  "id": "folder.abced3a35e6d1b",
  "from": {
    "name": null,
    "id": null
  },
  "name": "SkyDrive",
  "description": "",
  "parent_id": null,
  "size": 2957188732,
  "upload_location": "https://apis.live.net/v5.0/folder.abced3a35e6d1b/files/",
  "comments_count": 0,
  "comments_enabled": false,
  "is_embeddable": false,
  "count": 25,
  "link": "https://onedrive.live.com?cid=abced3a35e6d1b",
  "type": "folder",
  "shared_with": {
    "access": "Just me"
  },
  "created_time": null,
  "updated_time": "2014-06-13T18:00:54+0000",
  "client_updated_time": "2012-10-22T19:50:04+0000"
}
```

Figure 5 The `GetFolderItems` Method

```
public async Task<List<object>> GetFolderItems(string path, string filter)
{
    if (_session == null)
    {
        await InitProvider();
    }

    if (String.IsNullOrEmpty(path))
    {
        path = "me/skydrive";
    }

    if (!String.IsNullOrEmpty(filter))
    {
        filter = "?filter=" + filter;
    }

    LiveConnectClient client = new LiveConnectClient(_session);
    LiveOperationResult liveOpResult =
        await client.GetAsync(path + "/" + filter);
    dynamic dynResult = liveOpResult.Result;

    return new List<object>(dynResult.data);
}
```



TO BOLDLY CODE

WHERE NO VISUAL STUDIO LIVE!
HAS CODED BEFORE

WASHINGTON, D.C.

October 6 – 9, 2014 | Washington Marriott at Metro Center



**YOUR GUIDE TO THE
.NET DEVELOPMENT
UNIVERSE**

**Register by
August 13 and
SAVE \$300**

Topics include:

- Visual Studio/.NET Framework
- Windows Client
- JavaScript/HTML5 Client
- ASP.NET
- Cloud Computing
- Windows Phone
- Cross-Platform Mobile Development
- SharePoint
- SQL Server



Use promo code DCAUGTI



**BOLDLY CODE
WITH US!**

That's right. We are transporting Visual Studio Live! to our nation's capital for the first time in 21 years.

From Oct 6 – 9, 2014, developers, software architects, engineers and designers will gather for 4 days of cutting-edge education on the Microsoft Platform.

Join us on this special journey to explore topics covering all-things WCF, ALM, JavaScript, ASP.NET, Visual Studio and more!

Plus, you can explore this country's rich history just steps from the front door – all while sharpening your .NET skills and your ability to build better apps!

**Register by
August 13 and
SAVE \$300**



Use promo code **DCAUGTI**

CONNECT WITH VISUAL STUDIO LIVE!

 twitter.com/vslive – @VSLive

 facebook.com – Search "VSLive"

 linkedin.com – Join the "Visual Studio Live" group!

vslive.com/dc

SUPPORTED BY

Microsoft



Visual Studio

msdn
magazine

Visual Studio
MAGAZINE

PRODUCED BY



1105 MEDIA

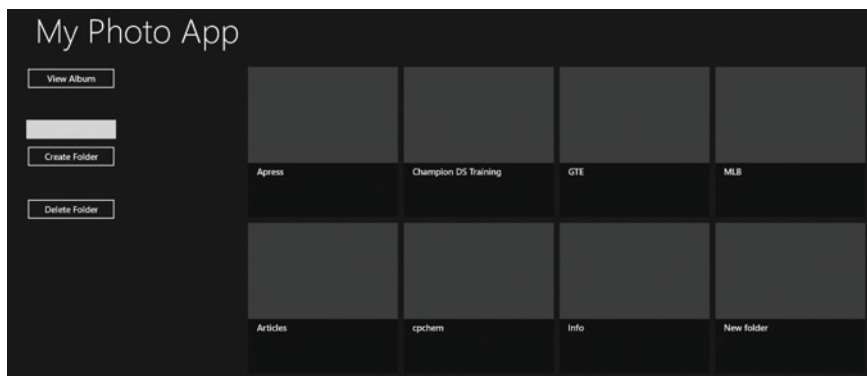


Figure 6 The FolderPage

types of objects I'll focus on in this article are: Folder, File, Album and Photo. You'll find a complete list of available objects with their description and use at msdn.microsoft.com/library/dn631843.

Once your app obtains an instance of `LiveConnectionSession` from the `LoginAsync` method of `LiveAuthClient`, that session instance can be used to create an instance of the `LiveConnectClient` class, which contains all of the API calls you'll use to interact with OneDrive. The following example shows creating an instance of the `LiveConnectClient` and querying OneDrive for the user's root folder object, which can be retrieved using the "me/skydrive" path:

```
public async Task<Object> GetRootFolder()
{
    LiveConnectClient client = new LiveConnectClient(_session);
    LiveOperationResult liveOpResult = await client.GetAsync("me/skydrive");
    dynamic dynResult = liveOpResult.Result;

    return dynResult;
}
```

This code uses the dynamic keyword to create a class instance of the underlying JSON-formatted string at run time. Though this is quick and easy to use, keep in mind there are some limitations you might run into that would require a more formal class structure.

The results returned from the `GetAsync` method will contain all of the properties for the requested object. **Figure 4** shows an example of the JSON-formatted string returned for a user's root folder.

When querying within OneDrive, it can be useful to create a more specific query. For instance, the following will return all child objects in the root directory:

```
LiveOperationResult liveOpResult = await client.GetAsync("me/skydrive/files");
```

Figure 7 The CreateFolder Method

```
public async Task<bool> CreateFolder(string path, string name)
{
    try
    {
        var folderData = new Dictionary<string, object>();
        folderData.Add("name", name);

        LiveConnectClient liveClient = new LiveConnectClient(_session);
        LiveOperationResult operationResult =
            await liveClient.PostAsync(path, folderData);
        dynamic result = operationResult.Result;
        return true;
    }
    catch (LiveConnectException exception)
    {
        return false;
    }
}
```

This includes folders and files and could potentially be a lot of data.

What if you want to see only a list of the photos in this directory? This can be accomplished using the filter query parameters. The OneDrive API supports several different types of query parameters in addition to filtering, such as limit, offset and search. The parameters allow you to get just the data you need, and they limit not only the amount of data that must be downloaded, but also the amount of code needed to handle the returned data.

For instance, the following query returns only a list of photos in the root directory, saving you from having to write code to separate out the photos from the folders:

```
LiveOperationResult liveOpResult =
    await client.GetAsync("me/skydrive/files?filter=photos");
```

A complete list of query parameters and their use can be found at msdn.microsoft.com/library/dn631842.

Working with Folders

If you're going to work with a file system of any sort, a logical place to start is with the folder structure. When querying for a folder, you can use two different types of identifiers: a friendly path like `me/skydrive` or a folder id that looks something like `folder:abcde86dfb3a35e6d1b.ABCDED3A35E6D1B!532`. If you want to query for a list of a folder's child objects, you can append `/files` to the path. For instance, the following will return a list of all files, folders, and albums for a given folder id:

```
LiveOperationResult liveOpResult =
    await client.GetAsync("folder.abcde86dfb3a35e6d1b.ABCDED3A35E6D1B!532/files");
```

OneDrive contains two types of folder objects: folder and album. An album can be considered a special type of folder found in a user's root OneDrive directory. Albums can contain photos and videos, as well as a folder structure and additional files.

It's important to understand the distinction between a folder and an album, because the following query would return only the folders in your root directory and ignore all of the albums. This might cause unexpected results:

```
LiveOperationResult liveOpResult =
    await client.GetAsync("me/skydrive/files?filter=folders");
```

To return both folders and albums, use a filter value of "folders,albums."

The previous query allows you to browse the entire folder structure of a user's OneDrive account by getting the child folders of each folder. The associated download for this article contains a sample app, `MyPhotoAlbum`, a XAML Windows Store app containing two pages. The first page, `FolderPage`, lets you browse your directory structure and shows the results in a `GridView`. All of the Live SDK calls are wrapped in the `OneDriveDataProvider` class.

Figure 5 shows the method used to traverse the folder structure by passing in the current folder id and using "folders,albums" as the filter value. If the folder id doesn't exist, the method defaults to the root directory.

Figure 6 shows an example of the results returned in the `FolderPage`. Selecting a child folder will navigate to the same `FolderPage`, using the child folder as a parameter. This allows the

navigation stack to preserve the back functionality to move back up the folder structure.

If you've requested update access using the `wl.skydrive_update` scope, you can also create and delete folders using the API. A folder contains only three properties you can create or change: name, description and `sort_by`. When creating a folder, you must specify a parent directory, either by path or id, and provide a JSON array with, at minimum, a name. Here's an example of the JSON needed for creating a new folder:

```
{
  "name": "My Favorite Photos",
  "description": "A folder full of my favorite photos."
}
```

The SDK handles this through the `PostAsync` method of the `LiveConnectClient`. **Figure 7** shows the method used to create a new folder.

Deleting a folder is done using the `DeleteAsync` method, which can also be used to delete files. It's important to remember about maintaining OneDrive's integrity and use any delete function with care:

```
LiveOperationResult operationResult = await liveClient.DeleteAsync(path);
```

Working with Files

While the `GetAsync` method will return a file's properties, it doesn't return the file itself. The only way to get a file from OneDrive is by downloading it. The Live SDK for Windows Store apps handles this by creating a download background task, which handles the download asynchronously while not bogging down the device. The following code can be used to download a file from OneDrive:

```
try
{
    LiveConnectClient liveClient = new LiveConnectClient(_session);
    LiveDownloadOperation op =
        await liveClient.CreateBackgroundDownloadAsync(filePath);
    var result = await op.StartAsync();
    // Handle result
}
catch
{
    // Handle errors
}
```

Once the background download operation is returned, it must be started by calling the `StartAsync` method of the `LiveDownloadOperation` instance.

Similar to downloading a file, the only way to add or update a file in OneDrive is to upload it. In order to upload a file, your app

Figure 8 Copying a File

```
public async Task<bool> CopyObject(string path, string destination)
{
    if (_session == null)
    {
        await InitProvider();
    }

    try
    {
        LiveConnectClient liveClient = new LiveConnectClient(_session);
        LiveOperationResult operationResult =
            await liveClient.CopyAsync(path, destination);

        return true;
    }
    catch (LiveConnectException exception)
    {
        return false;
    }
}
```

must have requested write access by using the `wl.skydrive_update` scope. The `CreateBackgroundUploadAsync` method takes a folder path, a filename, the stream containing the file and an overwrite option. If a file already exists, the overwrite option can either overwrite the original file or it can maintain the original file and rename the new file being uploaded. As with the folder update functions, take care when using this feature not to inadvertently destroy files. The following is an example of how to upload a file:

```
try
{
    LiveConnectClient liveClient = new LiveConnectClient(_session);
    LiveUploadOperation op = await liveClient.CreateBackgroundUploadAsync(
        path, filename, fileStream, OverwriteOption.Rename);
    var result = await op.StartAsync();
    // Handle result
}
catch
{
    // Handle errors
}
```

Before uploading a file, it's important to make sure the user has enough space for the new file. You can verify the amount of available space by using the path `"me/skydrive/quota"` in the `GetObjectAsync` method. This will return two properties, `quota` and `available`, which let you calculate the number of bytes remaining in the user's account.

There are two additional features in the API that come in handy when dealing with files. If you want to make a copy of a file, you could download the file and then upload it using a different name. However, that might use a lot of bandwidth for a rather simple operation. Moreover, what if you wanted to copy an entire folder, or move a child folder to a different parent folder? The code and bandwidth to do that using downloads and uploads would make such operations quite cumbersome. To address these functions, the OneDrive API has move and copy commands: `MoveAsync` and `CopyAsync`. Both take two parameters: the object—file or folder—to move and the destination path. **Figure 8** shows the method used to copy a file.

Wrapping Up

Only so much of an API can be covered in a single article. The Live SDK includes many more features, especially for dealing with photos and videos. If you're looking at integrating OneDrive into your solution, it would be very beneficial to explore these additional features. Microsoft has created the OneDrive Development Center to give you a single access point for everything related to the OneDrive API at dev.onedrive.com.

Whether you're building Windows Store apps, Windows Phone Store apps, or an app for any other platform that needs access to a user's file, adding OneDrive is a great way to integrate your app into a user's day-to-day life by granting him access to the files that matter most to him. So, as you build your great UX, be sure to include OneDrive. ■

TONY CHAMPION is president of *Champion DS*, is a Microsoft MVP, and is active in the community as a speaker, blogger, and author. He maintains a blog at tonychampion.net and can be reached via e-mail at tony@tonychampion.net.

THANKS to the following Microsoft technical expert for reviewing this article:
Mimi Sasouvanh



High Performance, Touch-Enabled UI Controls from DevExpress

Julian Bucknall, Chief Technical Officer

Become a **UI Superhero** and deliver elegant .NET solutions that fully address customer needs today and leverage your existing knowledge to build next-generation touch-enabled solutions for tomorrow. Whether it's an Office inspired application or a data centric analytics dashboard, DevExpress Universal ships with everything you'll need to build your best without limits or compromise.

Touch-Enabled UI for WinForms and WPF

With our most recent release, DevExpress continues its decade long commitment to .NET by extending its product line with enterprise-ready capabilities designed to help software developers build next-generation user experiences on the desktop platform of their choice. Our WPF and WinForms product lines empower developers and allow development teams to utilize existing skillsets and technology investments to meet tomorrow's challenges head-on.

"Deliver elegant line-of-business applications that emulate the touch-first Windows 8 Pro UX and maintain backward compatibility with previous versions of the Windows operating system."

Touch-Enabled UI for ASP.NET & MVC

"Take your applications where your users want to go and address the requirements of a BYOD world on the 2 most popular .NET web development platforms available today."

Because the technology landscape continues to evolve and new form factors emerge every day, the demands placed upon your ASP.NET and MVC web application has never been greater. Our web technologies allow you to create high-performance solutions for iPad, Surface or Android Tablets and effectively address the needs of next-generation apps by providing the touch-first interactivity users have come to expect from mobile operating systems and devices.

Windows 8 Store Ready Applications

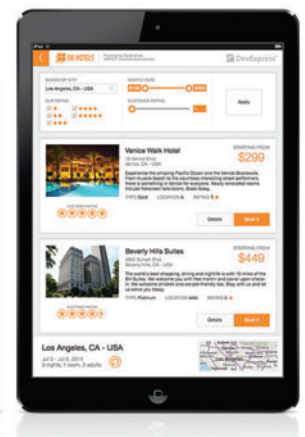
It's your choice: XAML or HTML5

"Create stunning desktop/tablet apps for the Microsoft Store with our Windows 8 XAML Controls or target Windows Phone mobile devices with DevExtreme."

When you are ready to move your solutions to Windows 8 and package and publish your products for sale on the Microsoft Store, our integrated suite of XAML 8 and JavaScript tools will help you create store-ready, multi-device applications with ease. From a rich collection of high performance XAML and HTML5 data visualization widgets to app-wide themes and screen layout tools... everything you'll need to build your best.

About DevExpress

DevExpress offers feature-complete UI controls, enterprise-ready reporting systems, IDE productivity tools and business application frameworks for Visual Studio. Our technologies help you build your best, see complex software with greater clarity, increase your productivity and create stunning touch-enabled applications for Windows, Web and next-generation Mobile platforms—without limits or compromise.



Feature-Complete .NET UI Controls by DevExpress

To access your 30-day, risk-free trial visit →

www.devexpress.com/trial

Developing Your First Game with Unity and C#

Adam Tuliper

As a software architect, I've written many systems, reverse-engineered native code malware, and generally could figure things out on the code side. When it came to making games, though, I was a bit lost as to where to start. I had done some native code graphics programming in the early Windows days, and it wasn't a fun experience. I then started on DirectX development but realized that, although it was extremely powerful, it seemed like too much code for what I wanted to do.

Then, one day, I decided to experiment with Unity, and I saw it could do some amazing things. This is the first article in a four-part series that will cover the basics and architecture of Unity. I'll show how to create 2D and 3D games and, finally, how to build for the Windows platforms.

This article discusses:

- The Unity interface
- Scenes, game objects and components
- Writing code
- Unity-generated code projects

Technologies discussed:

Unity, C#, Microsoft .NET Framework, Mono

Code download available at:

msdn.microsoft.com/magazine/msdnmag0814

What Unity Is

Unity is a 2D/3D engine and framework that gives you a system for designing game or app scenes for 2D, 2.5D and 3D. I say games and apps because I've seen not just games, but training simulators, first-responder applications, and other business-focused applications developed with Unity that need to interact with 2D/3D space. Unity allows you to interact with them via not only code, but also visual components, and export them to every major mobile platform and a whole lot more—for free. (There's also a pro version

Unity has done
some amazing work to ensure
cross-platform support.

that's very nice, but it isn't free. You can do an impressive amount with the free version.) Unity supports all major 3D applications and many audio formats, and even understands the Photoshop .psd format so you can just drop a .psd file into a Unity project. Unity allows you to import and assemble assets, write code to interact with your objects, create or import animations for use with an advanced animation system, and much more.

As **Figure 1** indicates, Unity has done work to ensure cross-platform support, and you can change platforms literally with one

click, although to be fair, there's typically some minimal effort required, such as integrating with each store for in-app purchases.

Perhaps the most powerful part of Unity is the Unity Asset Store, arguably the best asset marketplace in the gaming market. In it you can find all of your game component needs, such as artwork, 3D models, animation files for your 3D models (see Mixamo's content in the store for more than 10,000 motions), audio effects and full tracks, plug-ins—including those like the MultiPlatform toolkit that can help with multiple platform support—visual scripting systems such as PlayMaker and Behave, advanced shaders, textures, particle effects, and more. The Unity interface is fully scriptable, allowing many third-party plug-ins to integrate right into the Unity GUI. Most, if not all, professional game developers use a number of packages from the asset store, and if you have something decent to offer, you can publish it there as well.

Microsoft and Unity work closely together to ensure great platform support across the Microsoft stack.

What Unity Isn't

I hesitate to describe anything Unity isn't as people challenge that all the time. However, Unity by default isn't a system in which to design your 2D assets and 3D models (except for terrains). You can bring a bunch of zombies into a scene and control them, but you wouldn't create zombies in the Unity default tooling. In that sense, Unity isn't an asset-creation tool like Autodesk Maya or 3DSMax, Blender or even Adobe Photoshop. There's at least one third-party modeling plug-in (ProBuilder), though, that allows you to model 3D components right inside of Unity; there are 2D world builder plug-ins such as the 2D Terrain Editor for creating 2D tiled environments, and you can also design terrains from within Unity using their Terrain Tools to create amazing landscapes with trees, grass, mountains, and more. So, again, I hesitate to suggest any limits on what Unity can do.

Where does Microsoft fit into this? Microsoft and Unity work closely together to ensure great platform support across the Microsoft stack. Unity supports Windows standalone executables, Windows Phone, Windows Store applications, Xbox 360 and Xbox One.

Getting Started

Download the latest version of Unity and get yourself a two-button mouse with a clickable scroll wheel. There's a single download that

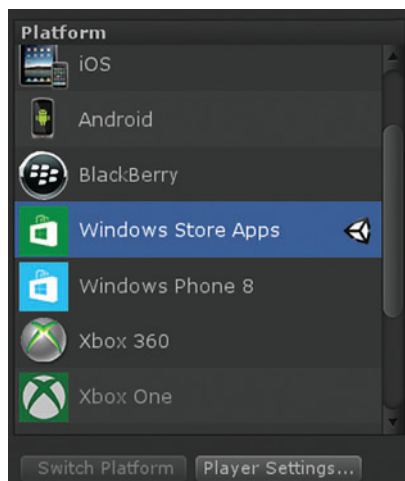


Figure 1 Platforms Supported by Unity

can be licensed for free mode or pro. You can see the differences between the versions at unity3d.com/unity/licenses. The Editor, which is the main Unity interface, runs on Windows (including Surface Pro), Linux and OS X.

I'll get into real game development with Unity in the next article, but, first, I'll explore the Unity interface, project structure and architecture.

Architecture and Compilation

Unity is a native C++-based game engine. You write code in C#, JavaScript (UnityScript) or, less frequently, Boo. Your code, not the Unity engine code, runs on Mono or the Microsoft .NET Framework, which is Just-in-Time (JIT) compiled (except for iOS, which doesn't allow

JIT code and is compiled by Mono to native code using Ahead-of-Time [AOT] compilation).

Unity lets you test your game in the IDE without having to perform any kind of export or build. When you run code in Unity, you're using Mono version 3.5, which has API compatibility roughly on par with that of the .NET Framework 3.5/CLR 2.0.

You edit your code in Unity by double-clicking on a code file in the project view, which opens the default cross-platform editor, MonoDevelop. If you prefer, you can configure Visual Studio as your editor.

You debug with MonoDevelop or use a third-party plug-in for Visual Studio, UnityVS. You can't use Visual Studio as a debugger without UnityVS because when you debug your game, you aren't debugging Unity.exe, you're debugging a virtual environment inside of Unity, using a soft debugger that's issued commands and performs actions.

To debug, you launch MonoDevelop from Unity. MonoDevelop has a plug-in that opens a connection back to the Unity debugger and issues commands to it after you Debug | Attach to Process in MonoDevelop. With UnityVS, you connect the Visual Studio debugger back to Unity instead.

When you open Unity for the first time, you see the project dialog shown in **Figure 2**.

In the project dialog, you specify the name and location for your project (1). You can import any packages into your project (2), though

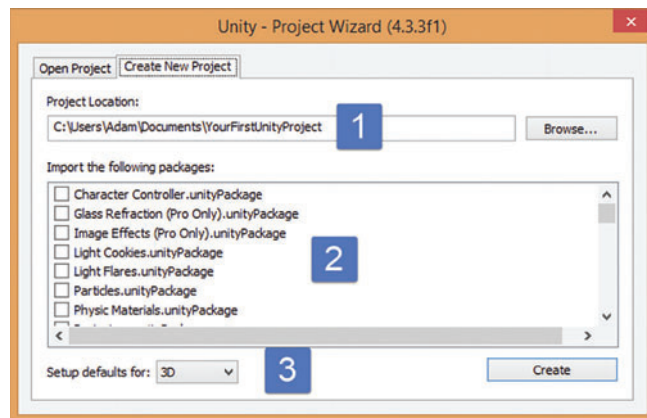


Figure 2 The Unity Project Wizard

you don't have to check anything off here; the list is provided only as a convenience. You can also import a package later. A package is a .unitypackage file that contains prepackaged resources—models, code, scenes, plug-ins—anything in Unity you can package up—and you can reuse or distribute them easily. Don't check something off here if you don't know what it is, though; your project size will grow, sometimes considerably. Finally, you can choose either 2D or 3D (3). This dropdown is relatively new to Unity, which didn't have significant 2D game tooling until fairly recently. When set to 3D, the defaults favor a 3D project—typical Unity behavior as it's been for ages, so it doesn't need any special mention. When 2D is chosen, Unity changes a few seemingly small—but major—things, which I'll cover in the 2D article later in this series.

This list is populated from .unitypackage files in certain locations on your system; Unity provides a handful on install. Anything you download from the Unity asset store also comes as a .unitypackage file and is cached locally on your system in C:\Users\<you>\AppData\Roaming\Unity\Asset Store. As such, it will show up in this list once it exists on your system. You could just double-click on any .unitypackage file and it would be imported into your project.

Continuing with the Unity interface, I'll go forward from clicking Create in the dialog in Figure 2 so a new project is created. The default Unity window layout is shown in Figure 3.

Here's what you'll see:

1. **Project:** All the files in your project. You can drag and drop from Explorer into Unity to add files to your project.
2. **Scene:** The currently open scene.
3. **Hierarchy:** All the game objects in the scene. Note the use of the term *GameObjects* and the GameObjects dropdown menu.
4. **Inspector:** The components (properties) of the selected object in the scene.
5. **Toolbar:** To the far left are Pan, Move, Rotate, Scale and in the center Play, Pause, Advance Frame. Clicking Play plays the game near instantly without having to perform separate builds. Pause pauses the game, and advance frame runs it one frame at a time, giving you very tight debugging control.
6. **Console:** This window can become somewhat hidden, but it shows output from your compile, errors, warnings and so forth. It also shows debug messages from code; for example, Debug.Log will show its output here.

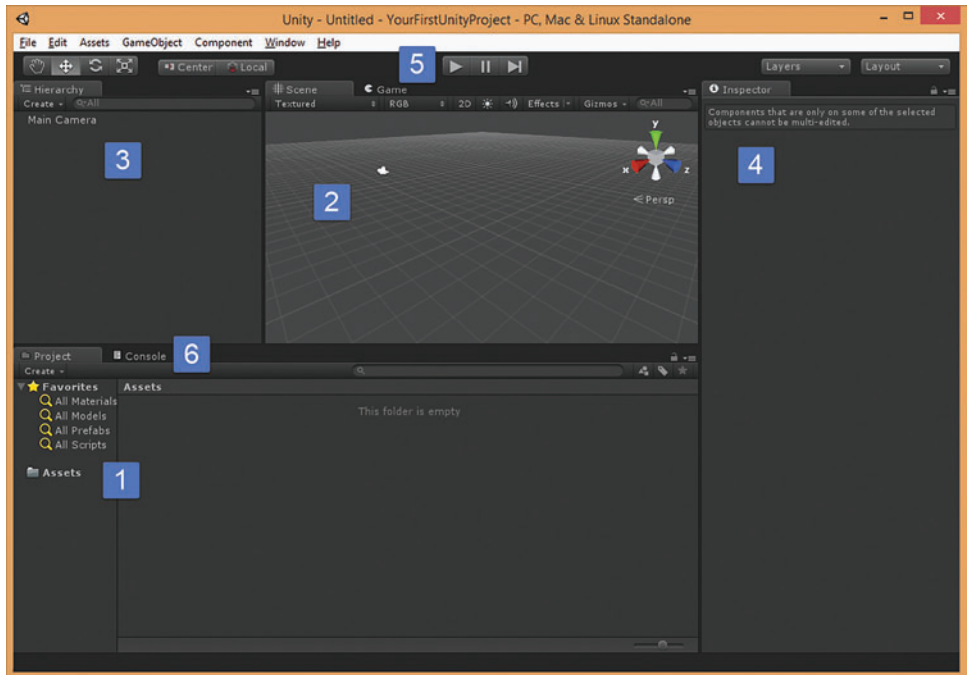


Figure 3 The Default Unity Window

Of important mention is the Game tab next to the Scene tab. This tab activates when you click play and your game starts to run in this window. This is called play mode and it gives you a playground for testing your game, and even allows you to make live changes to the game by switching back to the Scene tab. Be very careful here, though.

Everything that runs in your game exists in a scene.

While the play button is highlighted, you're in play mode and when you leave it, any changes you made while in play mode will be lost. I, along with just about every Unity developer I've ever spoken with, have lost work this way, so I change my Editor's color to make it obvious when I'm in play mode via Edit | Preferences | Colors | Playmode tint.

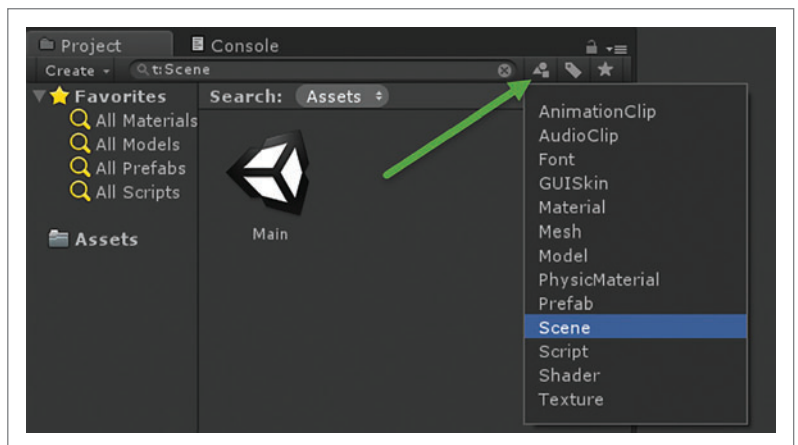


Figure 4 Filtering Scenes in the Project

Switch to Amyuni

For all your desktop and server PDF development needs.

Trusted, Proven Technology

Tired of using poorly-built PDF tools based on open-source libraries? Use the PDF technology installed on millions of desktops worldwide, say goodbye to open-source and get the reliability of fully supported software.

Hassle-free Integration and Distribution

Easily integrate powerful PDF functionality into your applications or servers with just a few lines of code. Quickly deploy your server or desktop based PDF solutions using our compact and light-weight printer driver and components.

Comprehensive Licensing

Whether you're a single developer or a corporation with a widely distributed application, you will find the right Amyuni license at the right pricing. From royalty-free to OEM licensing with minimal per-seat fee, we will adapt to your business needs and will not ask you to adapt to ours.

High-Performance

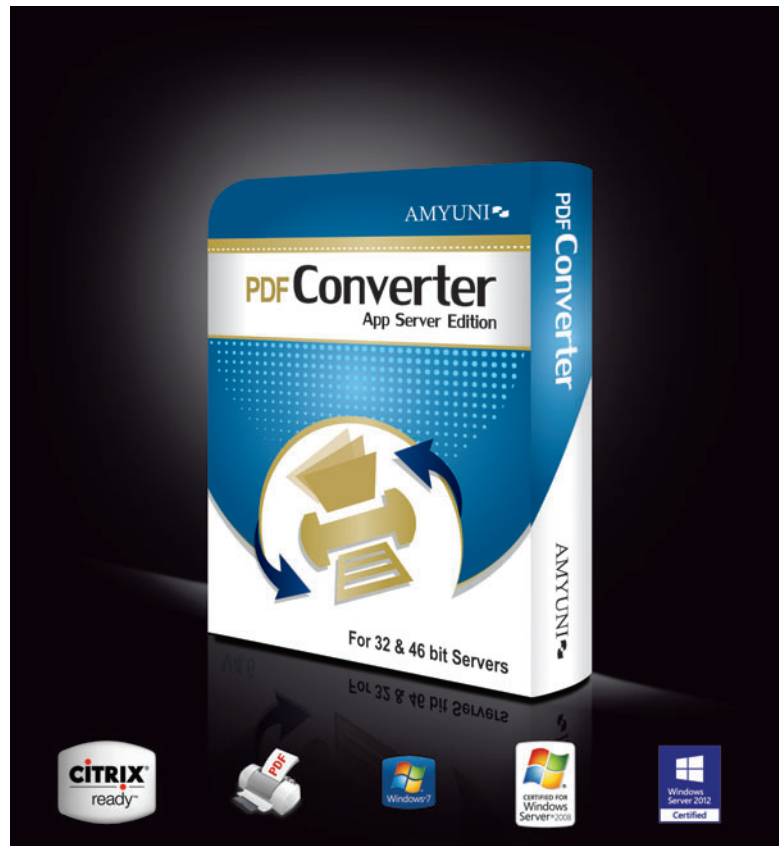
Develop with the fastest PDF conversion on the market, designed to perform in multithreaded and 64-bit Windows environments. Our PDF Converter is constantly benchmark tested and proven to be the top performer against all our competitors.

Hardware Certifications

From Windows XP to Windows 8.1, Server 2003 to Server 2012 R2, the Amyuni printer driver is WHQL certified and Citrix Ready! Available through Windows Updates, updating the Amyuni printer driver to a wide customer base could not be easier.

Fully Loaded Feature Set

Our expansive library of features is the result of providing custom PDF solutions for over 15 years. You will find most of the features provided by other libraries plus a number of features that you will find nowhere else.



Customizable to Suit your Needs

Choosing the right team of experts to help you with your implementation is key to your project success and meeting your deadlines. Owning our own technology means being able to quickly respond to your needs and adapt our libraries to new requirements.

We're Always a Phone Call Away

Do you have a concern or requirement that needs immediate attention? The Amyuni team is always here to help. By phone or by email, our team excels at providing the quickest response possible for your trickiest problems.

To learn more, please visit our website →

www.amyuni.com

To speak to one of our PDF experts, please call →

1-866-9AMYUNI

About Scenes

Everything that runs in your game exists in a scene. When you package your game for a platform, the resulting game is a collection of one or more scenes, plus any platform-dependent code you add. You can have as many scenes as you want in a project. A scene can be thought of as a level in a game, though you can have multiple levels in one scene file by just moving the player/camera to different points in the scene. When you download third-party packages or even sample games from the asset store, you typically must look for the scene files in your project to open. A scene file is a single file that contains all sorts of metadata about the resources used in the project for the current scene and its properties. It's important to save a scene often by pressing Ctrl+S during development, just as with any other tool.

Typically, Unity opens the last scene you've been working on, although sometimes when Unity opens a project it creates a new empty scene and you have to go find the scene in your project explorer. This can be pretty confusing for new users, but it's important to remember if you happen to open up your last project and wonder where all your work went! Relax, you'll find the work in a scene file you saved in your project. You can search for all the scenes in your project by clicking the icon indicated in **Figure 4** and filtering on Scene.

In a scene, you can't see anything without a camera and you can't hear anything without an Audio Listener component attached to some GameObject. Notice, however, that in any new scene, Unity always creates a camera that has an Audio Listener component already on it.

Project Structure and Importing Assets

Unity projects aren't like Visual Studio projects. You don't open a project file or even a solution file, because it doesn't exist. You point Unity to a folder structure and it opens the folder as a project. Projects contain Assets, Library, ProjectSettings, and Temp folders, but the only one that shows up in the interface is the Assets folder, which you can see in **Figure 4**.

The Assets folder contains all your assets—art, code, audio; every single file you bring into your project goes here. This is always

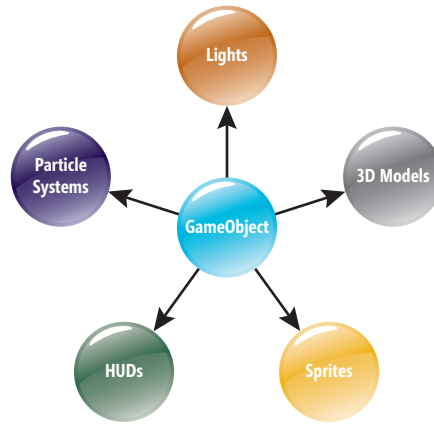


Figure 5 GameObjects in Unity

the top-level folder in the Unity Editor. But make changes only in the Unity interface, never through the file system.

The Library folder is the local cache for imported assets; it holds all metadata for assets. The ProjectSettings folder stores settings you configure from Edit | Project Settings. The Temp folder is used for temporary files from Mono and Unity during the build process.

I want to stress the importance of making changes only through the Unity interface and not the file system directly. This includes even simple copy and paste. Unity tracks metadata for your objects through the editor, so use the editor to

make changes (outside of a few fringe cases). You can drag and drop from your file system into Unity, though; that works just fine.

The All-Important GameObject

Virtually everything in your scene is a GameObject. Think of System.Object in the .NET Framework. Almost all types derive from it. The same concept goes for GameObject. It's the base class for all objects in your Unity scene. All of the objects shown in **Figure 5** (and many more) derive from a GameObject.

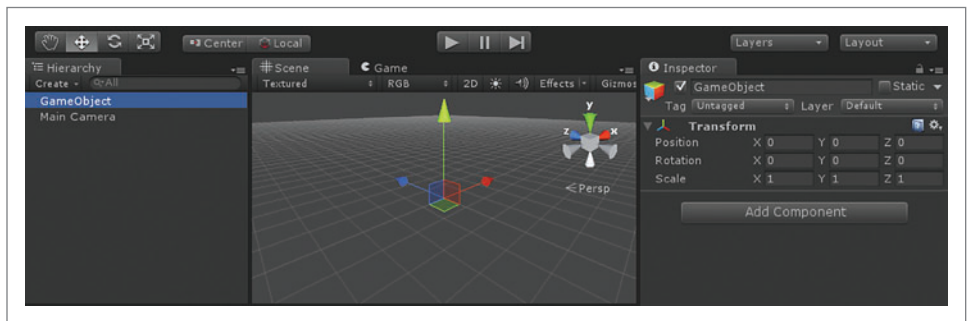


Figure 6 A Simple GameObject

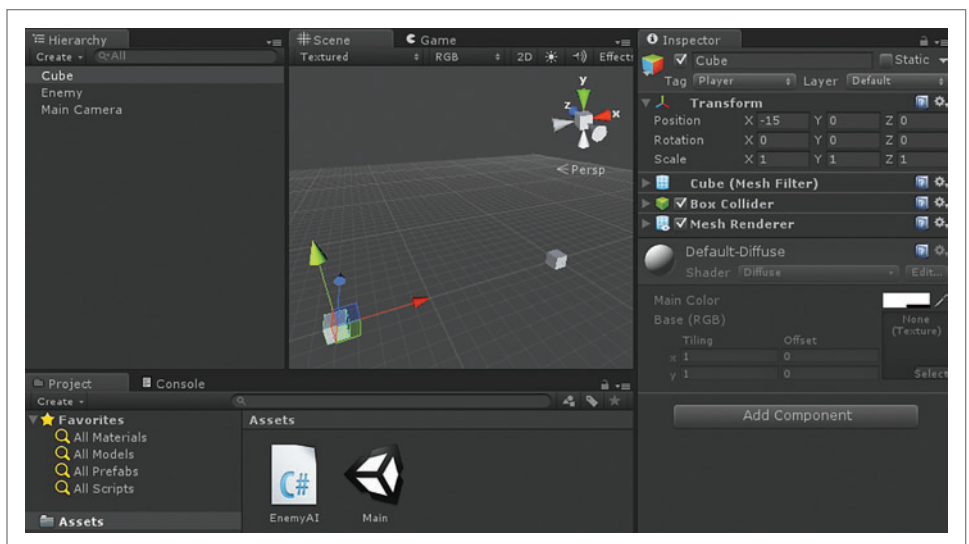


Figure 7 Current Project with Two Cubes

Creating a report is as easy as writing a letter



Reuse MS Word documents as your reporting templates



Create encrypted and print-ready Adobe PDF and PDF/A



Royalty-free WYSIWYG template designer



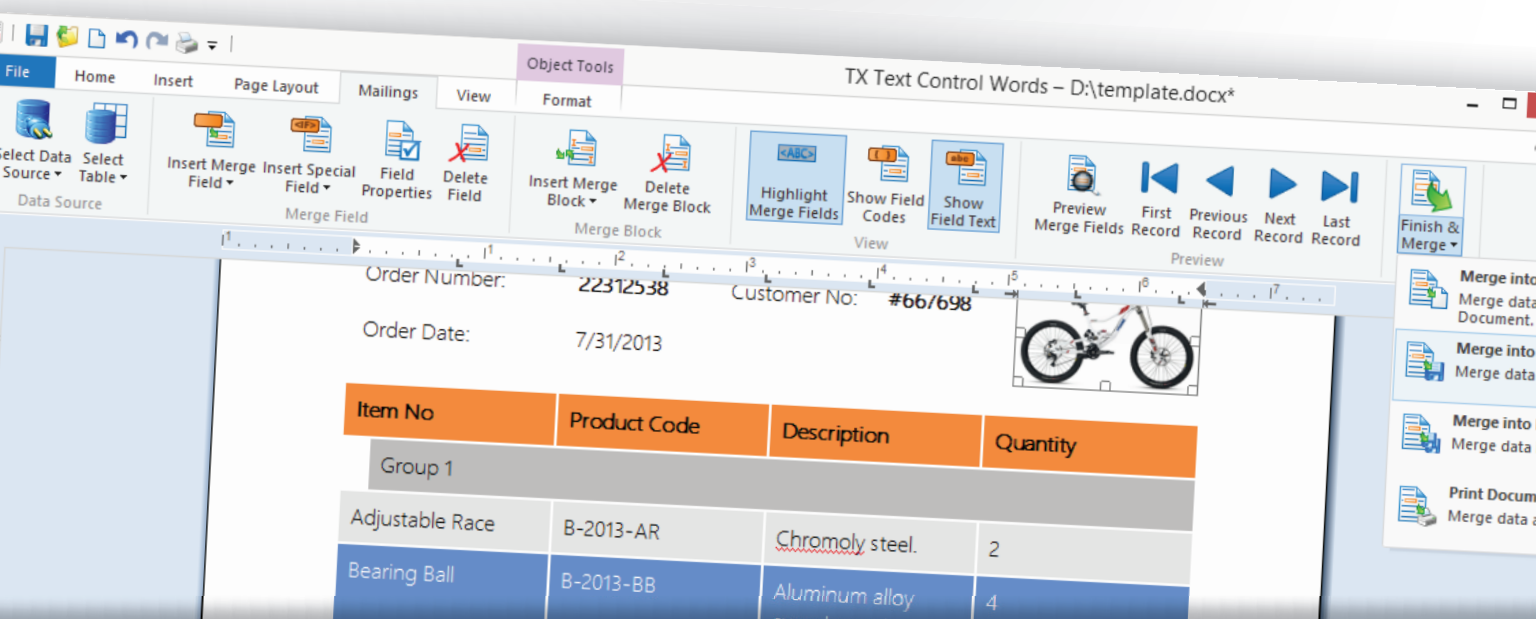
Powerful and dynamic 2D/3D charting support



Easy database connections and master-detail nested blocks



1D/2D barcode support including QRCode, IntelligentMail, EAN



www.textcontrol.com/reporting



txtextcontrol

US: +1 855-533-TEXT
EU: +49 421 427 067-10



Visual Studio

Microsoft

Partner

Reporting

Rich Text Editing

Spell Checking

Barcodes

PDF Reflow

A `GameObject` is pretty simple as it pertains to the Inspector window. You can see in **Figure 6** that an empty `GameObject` was added to the scene; note its properties in the Inspector. `GameObjects` by default have no visual properties except the widget Unity shows when you highlight the object. At this point, it's simply a fairly empty object.

A `GameObject` has a Name, a Tag (similar to a text tag you'd assign via a `FrameworkElement.Tag` in XAML or a tag in Windows Forms), a Layer and the Transform (probably the most important property of all).

The Transform property is simply the position, rotation and scale of any `GameObject`. Unity uses the left-hand coordinate system, in which you think of the coordinates of your computer screen as X (horizontal), Y (vertical) and Z (depth, that is, coming in or going out of the screen).

In game development, it's quite common to use vectors, which I'll cover a bit more in future articles. For now, it's sufficient to know that `Transform.Position` and `Transform.Scale` are both `Vector3` objects. A `Vector3` is simply a three-dimensional vector; in other words, it's nothing more than three points—just X, Y and Z. Through these three simple values, you can set an object's location and even move an object in the direction of a vector.

The Transform property is simply the position, rotation and scale of any `GameObject`.

Components

You add functionality to `GameObjects` by adding Components. Everything you add is a Component and they all show up in the Inspector window. There are `MeshRender` and `SpriteRender` Components; Components for audio and camera functionality; physics-related Components (colliders and rigidbodies), particle systems, path-finding systems, third-party custom Components, and more. You use a script Component to assign code to an object. Components are what bring your `GameObjects` to life by adding functionality, akin to the decorator pattern in software development, only much cooler.

I'll assign some code to a new `GameObject`, in this case a simple cube you can create via `GameObject | Create Other | Cube`. I renamed the cube `Enemy` and then created another to have two cubes. You can see in **Figure 7** I moved one cube about -15 units away from the other, which you can do by using the move tool on the toolbar or the W key once an object is highlighted.

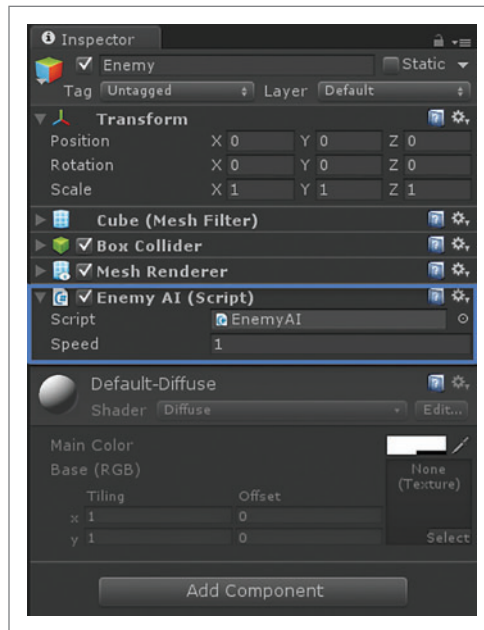


Figure 8 The Enemy with a Script Assigned to It

The code is a simple class that finds a player and moves its owner toward it. You typically do movement operations via one of two approaches: Either you move an object to a new position every frame by changing its Transform.Position properties, or you apply a physics force to it and let Unity take care of the rest.

Doing things per frame involves a slightly different way of thinking than saying “move to this point.” For this example, I'm going to move the object a little bit every frame so I have exact control over where it moves. If you'd rather not adjust every frame, there are libraries to do single function call movements, such as the freely available `iTween` library.

The first thing I do is right-click in the Project window to create a new C# script called `EnemyAI`. To assign this script to an object, I simply drag the script file from the project view to the object in

the Scene view or the Hierarchy and the code is assigned to the object. Unity takes care of the rest. It's that easy.

Figure 8 shows the `Enemy` cube with the script assigned to it.

Take a look at the code in **Figure 9** and note the public variable. If you look in the Editor, you can see that my public variable appears with an option to override the default values at run time. This is pretty cool. You can change defaults in the GUI for primitive types, and you can also expose public variables (not properties, though) of many different object types. If I drag and drop this code onto another `GameObject`, a completely separate instance of that code component gets instantiated. This is a basic example and it can be made more efficient by, say, adding a `Rigidbody` component to this object, but I'll keep it simple here.

In code, I can get a reference to any component exposed in the editor. I can also assign scripts to a `GameObject`, each with its own Start and Update methods (and many other methods). Assuming a script component containing this code needs a reference to the `EnemyAI` class (component), I can simply ask for that component:

```
public class EnemyHealth : MonoBehaviour

private EnemyAI _enemyAI;

// Use this for initialization.
void Start () {
    // Get a ref to the EnemyAI script component on this game object.
    var enemyAI = this.GetComponent<EnemyAI>();
}

// Update is called once per frame.
void Update () {
    _enemyAI.MoveTowardsPlayer();
}
```

After you edit code in MonoDevelop or your code editor of choice and then switch back to Unity, you'll typically notice a short delay. This is because Unity is background compiling your code. You can change your code editor (not debugger) via `Edit | Preferences | External Tools | External Script Editor`. Any compilation issues will show up at the very bottom status bar of your Unity Editor screen,



Do You Trust Your Data? Why Data Quality Matters.

Q&A with Bud Walker, Director of Data Quality Solutions

Q Who is Melissa Data, and what solutions do you provide?

A Founded in 1985, Melissa Data is a Rancho Santa Margarita, California company specializing in contact data quality solutions that power everything from bulk mailing software to instant identify verification in call centers and web forms. We verify address, name, phone and email info at point of entry or via batch processing. We can even match a name to an address for confidence that someone is who they say they are.

Q Why is contact data quality so important?

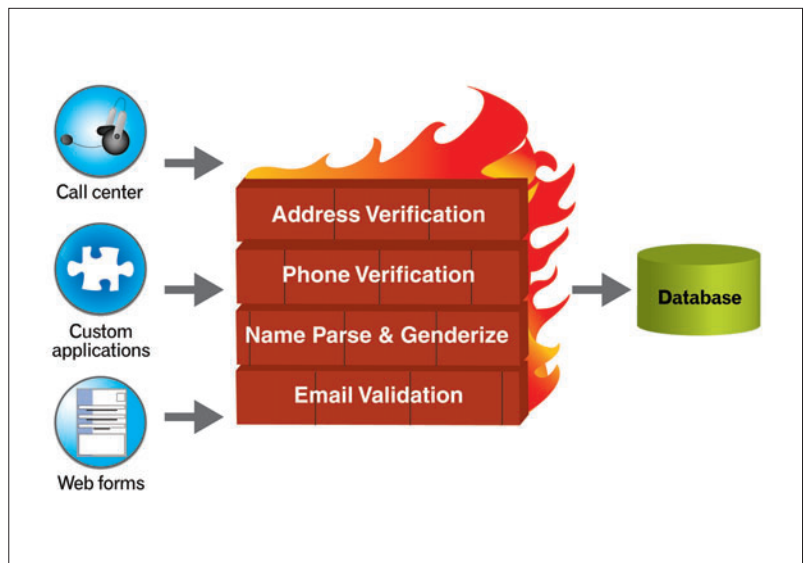
A Organizations rely on quality data for a variety of reasons—to power CRM initiatives; build customer loyalty; ensure invoices get paid; and help improve the accuracy of critical business decisions. Additionally, poor quality customer data costs U.S. businesses over \$600 billion dollars a year in postage, printing and staff overhead.

Q What are the typical causes of poor quality data?

A Data entry errors by employees and customers are the most common source of data defects reported, including misspellings, transpositions of numerals, and unrecognizable names. These types of errors are increasing as companies move their business to the Web and allow customers and suppliers to enter data about themselves directly into operational systems.

Q How do you see the industry evolving, what are the changing demands from clients that you are witnessing?

A We see changing demands from customers in a number of ways, including the need for tighter integration with enterprise systems. To meet this challenge we have continued a close partnership with Microsoft to integrate our toolset into SQL Server 2012—our tools are accessible through SQL Server Integration Services (SSIS) or via the Azure data market.



From a functional side, we are seeing greater demands for data profiling tools, and ways to utilize survivorship through record linkage and fuzzy matching to consolidate similar records into one golden record.

Demand for global data quality tools are escalating in demand—and we've met that demand with the roll out of a suite of global tools including global address verification, phone verification, data enrichment, and location geocoding.

Q How are your solutions typically employed?

A Our solutions are typically employed as APIs and Cloud services that need to be integrated into existing or custom applications, or as enterprise plugins to platforms such as SQL Server and ETL tools like Talend and Pentaho. We've seen greater migration to our Web services recently—as companies do not want to maintain and update the reference data themselves.

For more information or a free trial, please visit →

www.melissadata.com

Figure 9 The EnemyAI Script

```
public class EnemyAI : MonoBehaviour
{
    // These values will appear in the editor, full properties will not.
    public float Speed = 50;
    private Transform _playerTransform;
    private Transform _myTransform;

    // Called on startup of the GameObject it's assigned to.
    void Start()
    {
        // Find some gameobject that has the text tag "Player" assigned to it.
        // This is startup code, shouldn't query the player object every
        // frame. Store a ref to it.
        var player = GameObject.FindGameObjectWithTag("Player");
        if (!player)
        {
            Debug.LogError(
                "Could not find the main player. Ensure it has the player tag set.");
        }
        else
        {
            // Grab a reference to its transform for use later (saves on managed
            // code to native code calls).
            _playerTransform = player.transform;

            // Grab a reference to our transform for use later.
            _myTransform = this.transform;
        }

        // Called every frame. The frame rate varies every second.
        void Update()
        {
            // I am setting how fast I should move toward the "player"
            // per second. In Unity, one unit is a meter.
            // Time.deltaTime gives the amount of time since the last frame.
            // If you're running 60 FPS (frames per second) this is 1/60 = 0.0167,
            // so w/Speed=2 and frame rate of 60 FPS (frame rate always varies
            // per second), I have a movement amount of 2*0.0167 = .033 units
            // per frame. This is 2 units.
            var moveAmount = Speed * Time.deltaTime;

            // Update the position, move toward the player's position by moveAmount.
            _myTransform.position = Vector3.MoveTowards(_myTransform.position,
                _playerTransform.position, moveAmount);
        }
    }
}
```

so keep an eye out for them. If you try to run your game with errors in the code, Unity won't let you continue.

Writing Code

In the prior code example, there are two methods, Start and Update, and the class EnemyHealth inherits from the MonoBehaviour base class, which lets you simply assign that class to a GameObject. There's a lot of functionality in that base class you'll use, and typically a few methods and properties. The main methods are those Unity will call if they exist in your class. There are a handful of methods that can get called (see bit.ly/1jeA3UM). Though there are many methods, just as with the ASP.NET Web Forms Page Lifecycle, you typically use only a few. Here are the most common code methods to implement in your classes, which relate to the sequence of events for MonoBehaviour-derived classes:

Awake: This method is called once per object when the object is first initialized. Other components may not yet be initialized, so this method is typically used to initialize the current GameObject. You should always use this method to initialize a MonoBehaviour-derived class, not a constructor. And don't try to query for other objects in your scene here, as they may not be initialized yet.

Start: This method is called during the first frame of the object's lifetime but before any Update methods. It may seem very similar to Awake, but with Start, you know the other objects have been initialized via Awake and exist in your scene and, therefore, you can query other objects in code easily, like so:

```
// Returns the first EnemyAI script component instance it finds on any game object.
// This type is EnemyAI (a component), not a GameObject.
var enemyAI = GameObject.FindObjectOfType<EnemyAI>();
// I'll actually get a ref to its top-level GameObject.
var enemyGameObject = enemyAI.gameObject;
// Want the enemy's position?
var position = enemyGameObject.transform.position;
```

Update: This method is called every frame. How often is that, you ask? Well, it varies. It's completely computation-dependent. Because your system is always changing its load as it renders different things, this frame rate varies every second. You can press the Stats button in the Game tab when you go into play mode to see your current frame rate, as shown in **Figure 10**.

FixedUpdate: This method is called a fixed number of times a second, independent of the frame rate. Because Update is called a varying number of times a second and isn't in sync with the physics engine, it's typically best to use FixedUpdate when you want to provide a force or some other physics-related functions on an object.

FixedUpdate by default is called every .02 seconds, meaning Unity also performs physics calculations every .02 seconds (this interval is called the Fixed Timestep and is developer-adjustable), which, again, is independent of frame rate.

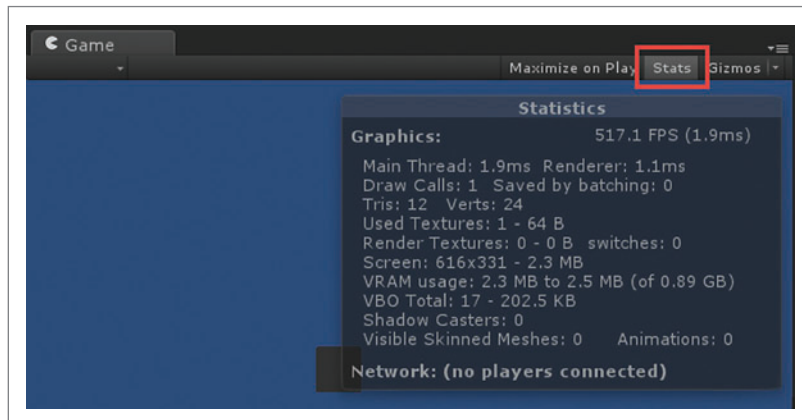


Figure 10 Getting Stats

Unity-Generated Code Projects

Once you have code in your project, Unity creates one or more project files in your root folder (which isn't visible in the Unity interface). These are not the Unity engine binaries, but instead the projects for Visual Studio or MonoDevelop in which you'll edit and compile your code. Unity can create what might seem like a lot of separate projects, as **Figure 11** shows, although each one has an important purpose.

PRECISELY PROGRAMMED FOR SPEED

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

ceTe Software's DynamicPDF products provide real-time PDF generation, manipulation, conversion, printing, viewing, and much more. Providing the best of both worlds, the object models are extremely flexible but still supply the rich features you need as a developer. Reliable and efficient, the high-performance software is easy to learn and use. If you do encounter a question with any of our components, simply contact ceTe Software's readily available, industry-leading support team.



DynamicPDF

[WWW.DYNAMICPDF.COM](http://www.DynamicPDF.com)



TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.631.5006 | +1 410.772.8620

ceTe software

If you have a simple Unity project, you won't see all of these files. They get created only when you have code put into various special folders. The projects shown in **Figure 11** are broken out by only three types:

- Assembly-CSharp.csproj
- Assembly-CSharp-Editor.csproj
- Assembly-CSharp-firstpass.csproj

For each of those projects, there's a duplicate project created with -vs appended to it, Assembly-CSharp-vs.csproj, for example. These projects are used if Visual Studio is your code editor and they can be added to your exported project from Unity for platform-specific debugging in your Visual Studio solution.

The other projects serve the same purpose but have CSharp replaced with UnityScript. These are simply the JavaScript (UnityScript) versions of the projects, which will exist only if you use JavaScript in your Unity game and only if you have your scripts in the folders that trigger these projects to be created.

Now that you've seen what projects get created, I'll explore the folders that trigger these projects and show you what their purposes are. Every folder path assumes it's underneath the /Assets root folder in your project view. Assets is always the root folder and contains all of your asset files underneath it. For example, Standard Assets is actually /Assets/Standard Assets. The build process for your scripts runs through four phases to generate assemblies. Objects compiled in Phase 1 can't see those in Phase 2 because they haven't yet been compiled. This is important to know when you're mixing UnityScript and C# in the same project. If you want to reference a C# class from UnityScript, you need to make sure it compiles in an earlier phase.

Phase 1 consists of runtime scripts in the Standard Assets, Pro Standard Assets and Plug-ins folders, all located under /Assets. This phase creates the Assembly-CSharp-firstpass.csproj project.

Phase 2 scripts are in the Standard Assets/Editor, Pro Standard Assets/Editor and Plug-ins/Editor folders. The last folder is meant for scripts that interact with the Unity Editor API for design-time functionality (think of a Visual Studio plug-in and how it enhances the GUI, only this runs in the Unity Editor). This phase creates the Assembly-CSharp-Editor-firstpass.csproj project.

Phase 3 comprises all other scripts that aren't inside an Editor folder. This phase creates the Assembly-CSharp-Editor.csproj project.

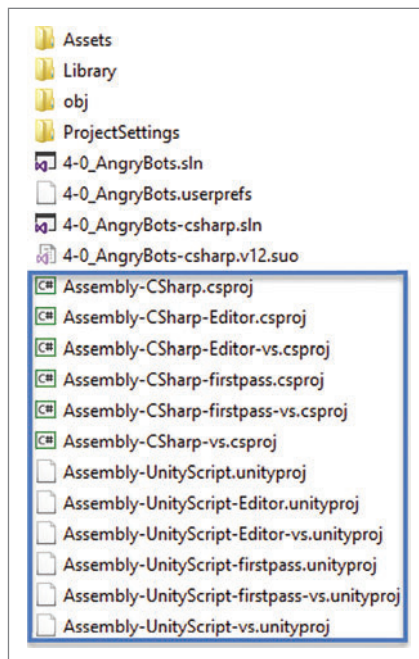


Figure 11 Unity-Created Projects

Phase 4 consists of all remaining scripts (those inside any other folder called Editor, such as /Assets/Editor or /Assets/Foo/Editor). This phase creates the Assembly-CSharp.csproj project.

There are a couple other less-used folders that aren't covered here, such as Resources. And there is the pending question of *what* the compiler is using. Is it .NET? Is it Mono? Is it .NET for the Windows Runtime (WinRT)? Is it .NET for Windows Phone Runtime? **Figure 12** lists the defaults used for compilation. This is important to know, especially for WinRT-based applications because the APIs available per platform vary.

When you perform a build for Windows, Unity is responsible for making the calls to generate the game libraries from your C#/UnityScript/Boo code (DLLs) and to include its native runtime libraries. For Windows Store and Windows Phone 8, it will export a Visual Studio solution, except for Windows stand-alone, in which Unity generates the .exe and

required .dll files. I'll discuss the various build types in the final article in the series, when I cover building for the platform. The graphics rendering at a low level is performed on the Windows platforms by DirectX.

Designing a game in Unity is a fairly straightforward process:

- Bring in your assets (artwork, audio and so on). Use the asset store. Write your own. Hire an artist. Note that Unity does have native support for Maya, Cheetah3d, Blender and 3dsMax, in some cases requiring that software be installed to work with those native 3D formats, and it works with .obj and .fbx common file formats, as well.
- Write code in C#, JavaScript/UnityScript, or Boo, to control your objects, scenes, and implement game logic.
- Test in Unity. Export to a platform.
- Test on that platform. Deploy.

But Wait, I Want More!

This article serves as an overview of the architecture and process in Unity. I covered the interface, basics of assigning code, GameObjects, components, Mono and .NET, plus more. This sets us up nicely for the next article where I'll dive right into assembling game components for a 2D game. Keep an eye on Microsoft Virtual Academy, as I'll be doing a two-day Unity learning event late summer. And watch for local regional learning events at unity3d.com/pages/windows/events. ■

Figure 12 Compilation Variations

Platform	Game Assemblies Generated By	Final Compilation Performed By
Windows Phone 8	Mono	Visual Studio/.NET
Windows Store	.NET	Visual Studio/.NET (WinRT)
Windows Standalone (.exe)	Mono	Unity - generates .exe + libs
Windows Phone 8.1	.NET	Visual Studio/.NET (WinRT)

ADAM TULIPER is a senior technical evangelist with Microsoft living in sunny Southern California. He's an indie game dev, co-admin of the Orange County Unity Meetup, and a pluralsight.com author. He and his wife are about to have their third child, so reach out to him while he still has a spare moment at adamt@microsoft.com or on Twitter at twitter.com/AdamTuliper.

THANKS to the following technical experts for reviewing this article: Matt Newman (Subscience Studios), Jaime Rodriguez (Microsoft) and Tautvydas Zilyus (Unity)



Big Data and Predictive Analytics Solutions from Syncfusion

Q&A with Daniel Lebaraj, Vice President of Syncfusion

Syncfusion, Inc. is a leading provider of .NET and JavaScript components, leveraging over 12 years of experience with Windows platforms and mobile devices. With over 10,000 customers, including many Fortune 500 companies with enterprise licenses, Syncfusion creates powerful frameworks that can answer any challenge, from trading systems to managing vast oil fields.

How can Syncfusion help developers with Big Data?

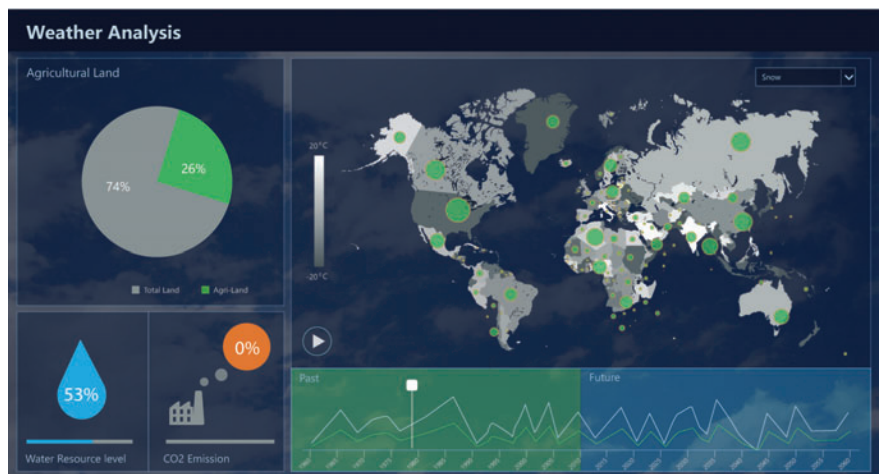
- Syncfusion takes the guesswork out of Hadoop, providing development-time support and tools for those working on the Windows platform.
- We provide the missing pieces to integrate big data solutions with .NET, using our tools on top of open source tools to simplify development.
- Easy-to-use installers are provided to enable one-click installation of a development-time Hadoop stack on any Windows machine. Check it out at Syncfusion.com/bigdatainstaller

Why Predictive Analytics?

- Predictive modeling solutions enable enterprises to use accumulated data to their advantage.
- Analyzing data can reveal which customers will pay on time, which will default, and other vital information.
- The down side is that until now, commercial modeling solutions have been very expensive. Additionally, the integration of modeling solutions with Windows-based .NET solutions has been complex.

How can Syncfusion help developers with Predictive Analytics?

- Our solutions combine the powerful R modeling environment with Syncfusion's own deployment framework to provide smooth integration with .NET.
- This enables customers to roll out powerful predictive analytics solutions.
- Solutions ship with an easy installer for Windows. Check it out at Syncfusion.com/predictiveanalyticsinstaller



Why choose Syncfusion?

- Syncfusion has over a decade of experience creating solutions for companies big and small.
- We have the expertise needed to make Big Data and Predictive Modeling work for customers on the Windows platform.
- We offer the unique ability to develop with the ease of Windows, and then deploy solutions to the Microsoft Azure cloud at a fraction of the price of comparable solutions.
- There are no per-node, per-user, or other deployment fees.

What kind of support does Syncfusion provide?

- Unlimited commercial support to enable customers to build Big Data and Predictive Modeling solutions.
- Samples, patches, and workarounds where applicable, all delivered to our SLA.
- Stress-free deployment to the Microsoft Azure cloud is guaranteed.

If you are not currently an enterprise customer, contact Syncfusion to find out more. Call 1-888-9DOTNET or email sales@syncfusion.com today!

To learn more, please visit our website →

www.syncfusion.com

Build MVVM Apps with Xamarin and MvvmCross

Thomas Lebrun

The Model-View-ViewModel (MVVM) pattern stands to become the reference pattern of choice for any XAML (Windows Presentation Foundation [WPF], Windows 8, Windows Phone and Silverlight) application. Introduced at the beginning of WPF, it separates concerns, testability and more. The best part is you can use it for any other technologies, even those that don't use XAML. Indeed, you can use the pattern with ASP.NET, with JavaScript and more.

Xamarin lets you develop Android or iOS applications in C# code. These applications come with their own development models, but thanks to a framework called MvvmCross, you can bring the MVVM pattern to these platforms as well. In this article, I'll give you all you need to understand MvvmCross and how to use it in your Android and iOS applications.

A Quick Look at MVVM

There have been plenty of articles covering MVVM lately, so I won't spend a lot of time reviewing the MVVM pattern. To summarize,

This article discusses:

- How to bring the MVVM pattern into other platforms such as iOS and Android
- Using Xamarin and C# to develop mobile apps for iOS and Android
- Developing code you can reuse across different mobile apps

Technologies discussed:

MvvmCross, Java, Microsoft .NET Framework

MVVM is composed of three parts: the Model (which corresponds to the data you'll want to display and manipulate on screen), the View (which is the presentation component and the UI) and the ViewModel (which will take the Model and display it on the View using data binding and will respond to user interaction). **Figure 1** shows a graphical representation of MVVM.

When developing with Microsoft technologies, it's easy to see the reusability provided by MVVM. But what about non-Microsoft technologies? What about Android? What about iOS?

Of course, you can still implement your own patterns or methodologies, but those might not provide some of the most powerful features of MVVM, such as data binding and testability. One of the biggest benefits of following MVVM is the ViewModels are easily testable. This also lets you push cross-platform code into the ViewModel. This code would otherwise be contained in a platform-specific class, like a controller.

Xamarin and MvvmCross “solve” this and offer a unified way to use MVVM on other platforms. Before looking at using MVVM on other platforms, I'll take a moment to explain Xamarin.

Xamarin for Android/iOS Applications

Xamarin is a set of tools that delivers high-performance compiled code with full access to all the native APIs. It lets you create native apps with device-specific experiences. Anything you can do in Objective-C or Java, you can do in C# with Xamarin.

While you can use Xamarin Studio to develop apps, you can also use Visual Studio and all the other tools you already use for C#

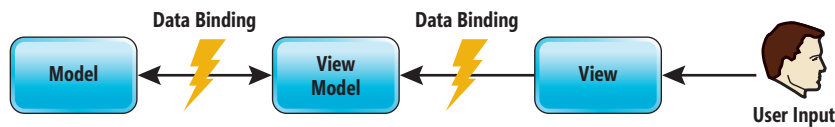


Figure 1 Overview of the Model-View-ViewModel Pattern

development today. This includes Team Foundation Server (for the source control) and plug-ins such as Resharper, GhostDoc and so on.

From a developer's perspective, Xamarin offers three main products—Xamarin.Mac, Xamarin.iOS (MonoTouch.dll) and Xamarin.Android (Mono.Android.dll). All of these are developed on top of Mono, the open source version of the Microsoft .NET Framework. Mono was actually initially created by Miguel De Icaza, the co-founder and current CTO of Xamarin.

In iOS, a dedicated compiler compiles applications written on C# directly to native ARM code. For Android, the process is similar to .NET compilation and execution. The sourcecode is compiled to an intermediate language (IL). When the code executes on the device, a second compilation (performed just in time) compiles the IL code to native code. This makes sense because Android applications are developed in Java, which has an internal architecture similar to the architecture of the .NET Framework. You can see a visual representation of the compilation process for iOS and Android in **Figure 2**.

Most of the time, you don't have to worry about the memory management, resource allocation and so on because everything is managed by the runtime Xamarin provides. However, there are cases when you have to be aware of what's happening, such as interop with Objective-C. This could create retain cycles or where your managed class actually wraps some expensive resources, such as UIImage on iOS. For more about this, see bit.ly/1iRCla2.

When developing with Microsoft technologies, it's easy to see the reusability provided by MVVM.

There are special considerations for iOS applications. While Xamarin Studio on a Mac provides everything needed for iOS development, Visual Studio users on a PC will still need a Mac with Xamarin tools installed. This lets you compile applications over the network and test them on the iOS Simulator or an iOS device.

Xamarin lets you build iOS and Android applications using C# or F#, but using the traditional Model-View-Controller pattern. If you want increased testability, maintainability and portability, you need a way to bring the MVVM pattern to these platforms. Enter MvvmCross.

MvvmCross for Xamarin Apps

MvvmCross is an open source, cross-platform MVVM framework developed by Stuart Lodge. It's available for Windows Phone, Windows 8, iOS, Android and WPF applications. MvvmCross brings the MVVM pattern to platforms where it was previously unavailable, like iOS and Android.

It also supports data binding in Views. This is a powerful feature that provides great separation of concerns. The View will use the ViewModels to offer proper behaviors in the application. MvvmCross even locates the ViewModels in a dedicated project so you can easily reference and reuse them in others.

This is the most important point when talking about MvvmCross. By locating the ViewModels in a Portable Class Library (PCL), you can add them as a reference to any other projects. Of course, that's not the only interesting point of MvvmCross. There's also a plug-in architecture, dependency injection (DI) and more.

Using MvvmCross on Android/iOS

Using MvvmCross is easy because it's only a few NuGet packages you add to your projects. Once you've done this, there are a few minor steps you have to take before launching the application. The steps vary a bit between iOS and Android, but they're quite similar. The Core project contains your ViewModels and the App class. This initializes the services and defines the ViewModel that will start on launch:

```
public class App : MvxApplication
{
    public override void Initialize()
    {
        this.CreatableTypes()
            .EndingWith("Service")
            .AsInterfaces()
            .RegisterAsLazySingleton();

        this.RegisterAppStart<HomeViewModel>();
    }
}
```

In your iOS or Android application, you have to create a Setup.cs file. This will reference the Core project and let the runtime know how to instantiate the application:

```
public class Setup : MvxAndroidSetup
{
    public Setup(Context applicationContext) : base(applicationContext)
    {
    }

    protected override IMvxApplication CreateApp()
    {
        return new Core.App();
    }
}
```

So the setup file creates the application using the app file. The latter one indicates to the runtime to load a particular ViewModel upon startup using the RegisterAppStart method.

Each view is specific to each application. This is the only part that changes. On Android, the class inherits from MvxActivity (standard activities on Android inherit from Activity). For iOS, the Views inherit from MvxViewController (standard ViewController on iOS inherit from UIViewController):

```
[Activity(ScreenOrientation = ScreenOrientation.Portrait)]
public class HomeView : MvxActivity
{
    protected override void OnViewModelSet()
    {
        SetContentView(Resource.Layout.HomeView);
    }
}
```

MvvmCross needs to know the ViewModel with which a View is associated. You can do this by default thanks to the naming

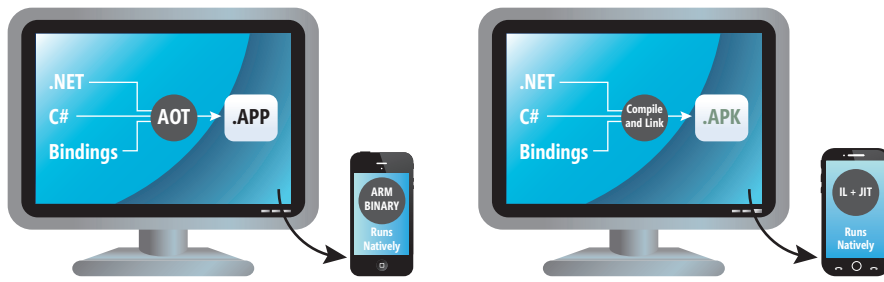


Figure 2 Native Compilation with Xamarin

convention. You can also easily change it by overriding the View-Model property of the View or using the `MvxViewForAttribute`:

```
[Activity(ScreenOrientation = ScreenOrientation.Portrait)]
[MvxViewFor(typeof(HomeViewModel))]
public class HomeView : MvxActivity
{ ... }
```

On iOS and Android, the Views are designed with different approaches. On iOS, the View is defined in C# code. On Android, you can also use C# code. Even better, though, you can use the AXML format (an XML format used to describe the UI on Android). Because of these differences, data bindings are defined differently on each platform.

On iOS, you create a `BindingDescriptionSet` to represent the link between the View and the ViewModel. In that set, you specify which control you want to bind to which property before applying the binding:

```
var label = new UILabel(new RectangleF(10, 10, 300, 40));
Add(label);
var textField = new UITextField(new RectangleF(10, 50, 300, 40));
Add(textField);

var set = this.CreateBindingSet<HomeView, Core.ViewModels.HomeViewModel>();
set.Bind(label).To(vm => vm.Hello);
set.Bind(textField).To(vm => vm.Hello);
set.Apply();
```

One of the biggest benefits
of following MVVM is the
ViewModels are easily testable.

On Android using AXML, you can use the new XML attribute `MvxBind` to perform the data binding:

```
<TextView xmlns:local="http://schemas.android.com/apk/res-auto"
    android:text="Text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tripitem_title"
    local:MvxBind="Text Name"
    android:gravity="center_vertical"
    android:textSize="17dp" />
```

The `MvxBind` attribute takes in parameters that specify the property of the control to bind and the property of the ViewModel to use as the source. If you're a XAML developer, be aware that the `MvvmCross` binding mode is `TwoWay` by default. In XAML, the default binding mode is `OneWay`. The `MvvmCross` Framework understands a few custom XML attributes available in `Mvx-BindingAttributes.xml`, as you can see in **Figure 3**.

The contents of this file are simple, but very important. The file indicates the attributes you can use in the AXML files. So, you can see in a binding operation, you can use `MvxBind` or `MvxLang` attributes. You can also use some new controls (`MvxImageView`, `MvxListView`). Each of them has dedicated custom attributes, as you can see in **Figure 4**.

This should be familiar to XAML developers. The `ItemsSource` and `ItemClick`

properties are bound to some properties of the data source (the ViewModel in this case). The `MvxItemTemplate` defines the interface for each item in the ListView.

You might expect the syntax for iOS to be quite different, but, in reality, it's actually quite similar. The syntax used in the AXML file, referred to as "Fluent" binding, is simply a text-format binding you can map to the C# version as well. The command used in the previous example to select an item on the list is an `ICommand` object:

```
public ICommand<Trip> SelectTripCommand { get; set; }
```

The implementation of this interface is provided by `MvvmCross` using the `MvxCommand` class:

```
private void InitializeCommands()
{
    this.SelectTripCommand = new MvxCommand<Trip>(
        trip => this.ShowViewModel<TripDetailsViewModel>(trip),
        trip => this.Trips != null && this.Trips.Any() && trip != null);
}
```

A common problem when using the MVVM pattern is converting types. This happens when you define properties using a type that isn't directly consumable by the UI. For example, you might have an image property as a byte array but you want to use it for the source property of an image control. In XAML, you can solve this problem with the `IValueConverter` interface, which maps values between the View and the ViewModel.

The process is quite similar with `MvvmCross`, thanks to the `IMvxValueConverter` interface and its two methods `Convert` and `ConvertBack`. This interface lets you perform conversions similar to XAML, but with a cross-technology objective in mind. Keep in mind this interface has the same drawback as XAML. It takes an object as a parameter and returns it as the value. So casting is required. To optimize this, `MvvmCross` offers the generic `Mvx-ValueConverter` class, which takes in parameters for the input and the return types:

```
public class ByteArrayToImageConverter : MvxValueConverter<byte[], Bitmap>
{
    protected override Bitmap Convert(byte[] value, Type targetType,
        object parameter, CultureInfo culture)
    {
        if (value == null)
            return null;

        var options = new BitmapFactory.Options { InPurgeable = true };
        return BitmapFactory.DecodeByteArray(value, 0, value.Length, options);
    }
}
```

Referencing the converter is easy. In iOS, use the method `WithConversion` in the fluent syntax:

```
var set = this.CreateBindingSet<HomeView, Core.ViewModels.HomeViewModel>();
set.Bind(label).To(vm => vm.Trips).WithConversion("ByteArrayToImage");
set.Apply();
```




ReSharper: Turbocharge Your .NET Development

A Visual Studio 2013 Q&A with Sergey Shkredov,
Development Department Lead of JetBrains

Q What is JetBrains ReSharper?

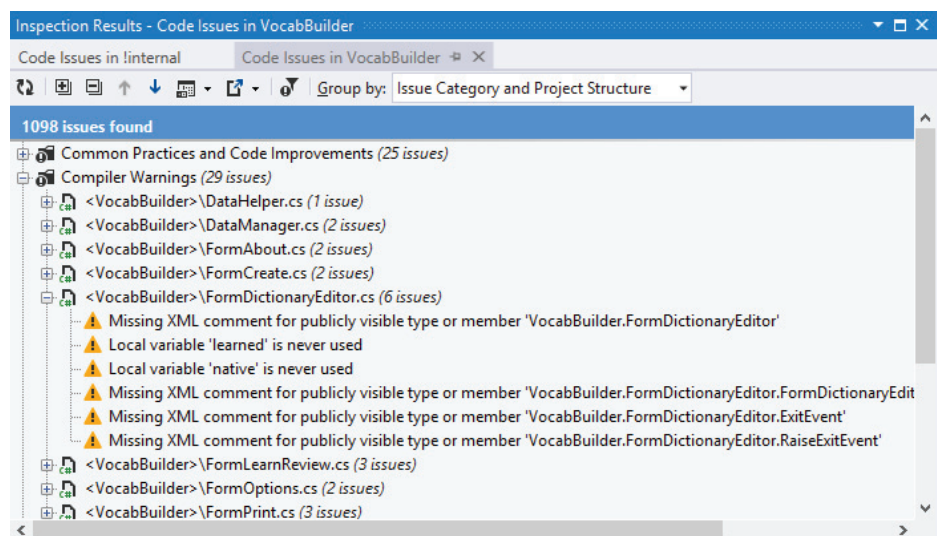
A ReSharper is a popular extension to Visual Studio. Over 220,000 .NET developers worldwide rely on ReSharper to be more productive in creating and improving code in C#, VB.NET, ASP.NET, JavaScript, TypeScript, CSS, HTML, XML, and XAML. It is available in multiple versions of Visual Studio, up to version 2013.

Q How does ReSharper manage to stay so popular, even as Visual Studio improves with each release?

A We're improving, too! Compared to barebones Visual Studio, ReSharper warns developers of many more types of coding problems, provides more comprehensive tools for code maintenance, and brings better ways of reading and understanding code. We also invest in making developers feel at home with more languages used in the current .NET infrastructure, such as JavaScript and TypeScript.

Q What's best about ReSharper for developers?

A ReSharper automates so many things in the development process. It finds errors, redundancies, code smells, and possible improvements right as you type, suggesting intelligent corrections. It also helps you explore code by visualizing file structure, class hierarchies, call and value chains, and project dependencies. You can instantly traverse your entire solution and jump right to the exact file and line that you are looking for; ReSharper will even decompile library code if necessary. Dozens of solution-wide refactorings are available to help you safely change your code base. Add code formatting, unit testing, internationalization, and smart IntelliSense, and you've got yourself a must-have tool.



Q Does ReSharper provide benefits for the business?

A Oh, absolutely. For one, ReSharper reveals problems in code early in the application development process. This includes detection of possible exceptions and bugs that could surface during application execution; possibly inadequate architecture; usage of dubious language constructs; or inaccurate code logic, just to name a few. As a result, you can spot bad code at a very early stage and thus eliminate bugs in the cheapest way possible.

Another example is that ReSharper saves time and effort required to maintain coding standards across the team. ReSharper's code inspections help verify (and, if necessary, fix) names of code symbols, as well as enforce teamwide code formatting settings. Think about the time your team would otherwise waste trying to make code compliant to team guidelines. ReSharper saves this time for you.

To learn more about ReSharper and
download a free 30-day trial, please visit →

www.jetbrains.com/msdn

Figure 3 Contents of the MvxBindingAttributes.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <declare-styleable name="MvxBinding">
    <attr name="MvxBind" format="string"/>
    <attr name="MvxLang" format="string"/>
  </declare-styleable>
  <declare-styleable name="MvxControl">
    <attr name="MvxTemplate" format="string"/>
  </declare-styleable>
  <declare-styleable name="MvxListView">
    <attr name="MvxItemTemplate" format="string"/>
    <attr name="MvxDropDownItemTemplate" format="string"/>
  </declare-styleable>
  <item type="id" name="MvxBindingTagUnique">
    <declare-styleable name="MvxImageView">
      <attr name="MvxSource" format="string"/>
    </declare-styleable>
  </item>
</resources>
```

Figure 4 New Controls Have Custom Attributes

```
<LinearLayout
  android:orientation="horizontal"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:layout_weight="2">
  <Mvx.MvxListView
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    local:MvxBind="ItemsSource Trips;ItemClick SelectTripCommand"
    local:MvxItemTemplate="@layout/tripitemtemplate" />
  </LinearLayout>
```

In Android, reference the converter directly in the AXML file:

```
<ImageView
  local:MvxBind="Bitmap Image,Converter=ByteArrayToImage"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content" />
```

Converters are located by their names using reflection. By default, the framework will search for any type containing “converter” in the name. You can also manually register converters by overriding the `FillValueConverters` method in the `Setup` class.

Anything you can do in
Objective-C or Java, you can do
in C# with Xamarin.

MvvmCross provides a simple and light `DI` container. You can register classes and interfaces in the container using multiple patterns including a singleton registration, a dynamic registration and more:

```
Mvx.RegisterType<ISQLiteConnectionFactory, SQLiteConnectionFactory>();
Mvx.RegisterSingleton<ISQLiteConnectionFactory, SQLiteConnectionFactory>();
```

Resolving types in the container can happen two ways. First, you can use the `Mvx.Resolve` method to explicitly resolve the type. It also supports constructor injection, which lets MvvmCross perform reflection and automatically resolve parameters during object creation:

```
private readonly ISQLiteConnectionFactory _sqlFactory;
public DataAccessLayerService(ISQLiteConnectionFactory sqlFactory)
{
  this._sqlFactory = sqlFactory;
}
```

You can use constructor injection for services, as well as View-Models. This is important to understand because any services developed for your application put in the Core project are, by default, cross-platform.

You can also leverage services that seem to be cross-platform, but for which implementation is platform-specific. For example, taking a picture with the camera, getting the user coordinates, using a database and so on. With constructor injection, ViewModels can receive an interface where the implementation is platform-specific.

MvvmCross brings the MVVM
pattern to platforms where
it was previously unavailable,
like iOS and Android.

To further define this mechanism of injection and platform-specific code, MvvmCross provides a plug-in system. The system lets you create and inject new features at run time. Each plug-in is a service, exposed by an interface, which has a concrete implementation provided for each platform. The plug-in system registers the interface and implementation. Applications consume the plug-ins with DI. DI also lets plug-in developers provide a simplified “mock” implementation during testing and development.

From a developer’s point of view, writing a plug-in is as simple as writing an interface. You create a class that implements that interface and a plug-in loader. The plug-in loader is a class that implements the `IMvxPluginLoader` interface. It registers the plug-in interface and implementation (using `Mvx.RegisterType`) when its `EnsureLoaded` method is called.

There are many plug-ins already available. They’ll provide features such as file access, e-mail, JSON conversion and so on. You can find most of them using NuGet. Be aware that some plug-ins don’t include implementations for all platforms. Pay attention to this detail when planning to leverage a plug-in. Even if a plug-in is missing support for your platform, you may find it easier to follow the pattern and implement the missing platform instead of creating a new plug-in on your own. If this happens, consider contributing your implementation back to the plug-in owner so others can also use it as well.

MvvmCross is a valuable framework. Consider this when developing mobile applications—even on Android and iOS. The MVVM pattern, coupled with data binding and plug-ins, provides a powerful system for creating highly maintainable and portable code. ■

THOMAS LEBRUN is a consultant at Infinite Square, a French Microsoft partner working on technologies such as Windows 8, Windows Phone, Windows Presentation Foundation (WPF), Silverlight, Surface and more. He has written two books about WPF and the MVVM pattern. He’s also a regular speaker at community events. You can follow him on his blog at blog.thomaslebrun.net and on Twitter at twitter.com/thomas_lebrun.

THANKS to the following Microsoft technical expert for reviewing this article:
Jared Bienz



Talking Scrum with the Founder of Axosoft

Founder Hamid Shojaei has guided Axosoft to becoming the #1 Scrum software for developers by focusing on the core features Scrum teams need to ship great software on time and on budget.



Q With the #1 Scrum product and video, it's clear Axosoft is really invested in this framework. Why did you decide to go "all in" on Scrum?

A Because Scrum works. It's that simple. Scrum helps deliver software faster and better than any methodology which is why we have wholeheartedly embraced it. Plus, we appreciate the simplicity and adaptability of its core concepts. As the makers of a Scrum tool, we are always trying to integrate these traits into our product so we have created a straightforward, one-page interface that also allows teams to customize fields and create flexible workflows.

Q What are some of the new things Axosoft is doing to support the Scrum community?

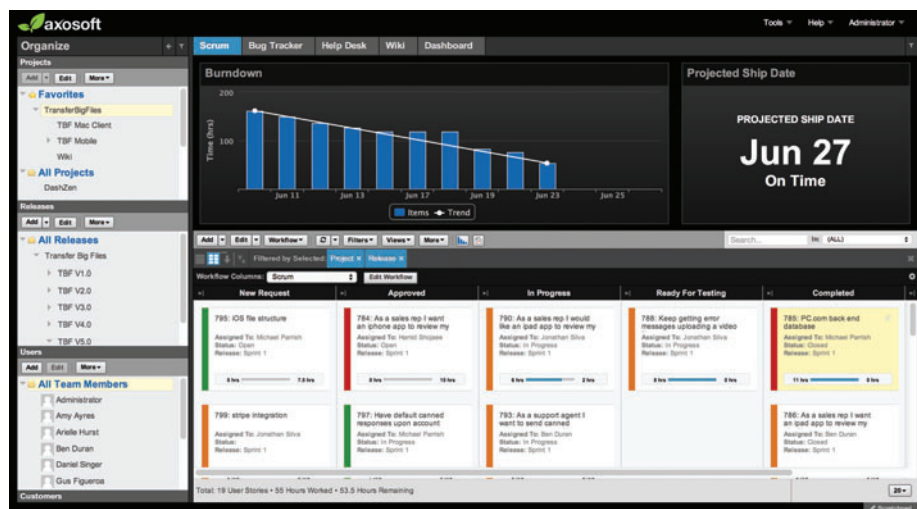
A In addition to hosting local Scrum User Groups in our office, we're really proud of the traction our new Scrum education site, ScrumHub.com, has gained. We believe Scrum concepts should be presented in an easy to learn way, so we've made a bunch of awesome free guides, webinars, videos and infographics that illustrate the fundamentals.

Q Aside from the Scrum piece, your company also recently decided to give away the Axosoft Bug Tracker. Why did you decide to do this?

A There are two big reasons why we decided to give away Axosoft Bug Tracker: First off, we wanted to improve every dev team's ability to track defects and build better software—even when budgets are tight. The second reason is a little selfish, but we also wanted to introduce Axosoft to the world! We expect that the people who use Axosoft Bug Tracker will love it and consider checking out our other products too.

Q Axosoft has been consistently voted one of the best places to work in Arizona. How do you develop an exceptional workplace?

A I just try to create the kind of work environment I would want to work in! After all, I'm going to spend most of my life at work, so I want to surround myself with smart people who can have intelligent conversations and a little bit of fun too. That's pretty much the foundation of every work environment decision we make, and it helps us bring on even more great people.



Q What are Axosoft's plans for the future?

A As a company our purpose is to help dev teams ship software on time and on budget. Though that's a really simple statement, there's a lot we can keep doing to improve the way developers work. We will continue making enhancements to the Axosoft product suite to achieve that singular purpose.

A recent example of this would be our new "Daily Scrum" feature that helps dev teams conduct daily Scrum meetings more efficiently by showing the work the team has done in the last day.

To try Axosoft Scrum for free or sign up for Axosoft Bug Tracker, please visit



www.axosoft.com

Microsoft Azure Media Services

Gregory Prentice

Over the years, Microsoft has helped design and support large-scale, live streaming video delivery. One prime example is the 2014 Sochi Olympics. This required a huge amount of technological resources, including hardware servers, source video streams, encoding, server redundancy (dual datacenters), output-adaptive video streams, Content Delivery Networks (CDNs), partner companies and, of course, Microsoft staff.

This article discusses:

- How to use Microsoft Azure Media Services to orchestrate the components of a live video stream
- Which technological components are required to process and deliver live video streams
- How to best configure and coordinate video streams for viewing on multiple device types

Technologies discussed:

Microsoft Azure Media Services, Microsoft Azure, Visual Studio 2012

Code download available at:

msdn.microsoft.com/magazine/msdnmag0814

BUILD A DEV/TEST SANDBOX IN THE CLOUD FOR FREE

MSDN subscribers can quickly spin up a dev/test environment on Microsoft Azure at no cost. Get up to \$150 in credits each month!

aka.ms/msdnmag

All these resources—along with customer software development—are needed to ensure the end-to-end live video event is executed with few or no incidents. The cost of orchestrating these resources is reflective of the scale of the live streaming event. There's a high capital expense incurred to purchase the required servers, switches and related technology. This leads to obvious risks and the following questions:

- Was enough or too much hardware purchased to handle the scale of the live event?
- What do you do with said hardware until the next event?
- How long will the acquired technology still be relevant?
- Were the correct technology partners included to ensure the event is orchestrated flawlessly?

Considering Microsoft's involvement, knowledge and success in live video events, the Microsoft Azure Media Services team developed live streaming to mitigate these risks. The team successfully streamed live global video feeds of the 2014 Sochi Olympics, running on Microsoft Azure Media Services. While that was an extremely large event, Microsoft Azure provides on-demand hardware to address scalability.

Depending on the size of the live video event, a request to Azure Media Services may create a scale unit—small, medium or large—to provide streaming live video to hundreds of thousands or just hundreds of viewers. The pricing allows pay-as-you-go models to procure, use and release the live streaming service, providing the costs are known prior to an event. The hardware and infrastructure are continually upgraded and refreshed. Microsoft

also maintains strategic relationships that are relevant to delivering live streaming solutions.

In this article, I'll focus on an example scenario that will use the new live streaming from the Microsoft Azure Media Services team (currently in private preview), address the aforementioned risks and discuss the relationship of the live streaming offering to the existing video-on-demand (VOD) service.

Concepts and Terminology

It's important to understand some general concepts and terminology regarding Azure Media Services. The term "channel" refers to the developer's view of the full end-to-end path that one live video stream takes through Azure Media Services. A channel can be in various states, with the two most important being "stopped" and "running." A channel includes the following components that coordinate the video streams through the system:

- **Ingest URI:** Represents an ingress point at which the channel receives one or more video bitrate streams for delivery.
- **Preview URI:** Represents an egress point of the live stream as received by the channel, which should only be used for monitoring purposes.
- **Program:** Associated with a channel and represents a portion of the live stream persisted to Blob storage as an asset. At least one program should be created on a channel and in a running state to enable a live video stream. The program will actively capture video while in a running state.
- **Asset:** Associated with a program, this represents data stored in Blob storage. An Asset might contain one or more files including video, audio, images, thumbnail collections and manifest. The asset may still exist even after deleting a program.
- **Locator:** Associated with the asset and an origin. A locator provides a URI used as the egress point of the live program stream.
- **Origin:** Associated with a locator, this represents the scalable egress point for the video streams' delivery from an asset's locator URI in multiple bitrate (MBR) formats, such as Smooth, HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP.

Azure Media Services ensures proper redundancy is created and available for reliable delivery of video at scale. That delivery could involve many devices that consume varying streaming formats, such as Smooth, HLS and DASH. **Figure 1** shows what constitutes a channel.

It's important to understand the relationship between a channel and one or more programs created within the channel. In **Figure 2**, a live video event has been mapped onto a channel representation that's in a running state from 5 p.m. until 2 a.m. It's important to note the channel's transition time from a stopped state to a running state can require 10 to 20 minutes. The interval markers—Concert1, Vip1, Act1, Act2 and Vip2—represent programs and their planned start and stop times. Finally, the interval markers named Transition represent the time between the current program and the next program, where one is stopped and the other is started.

Microsoft Azure Media Services ensures proper redundancy is created and available for reliable delivery of video at scale.

This is one example of how a timeline might map onto a channel using programs. As defined in **Figure 2**, there's one program that spans the entire event.

Concert1 will be in a running state from 5:15 p.m. until 2 a.m., with a sliding video buffer of 10 minutes. The buffering lets users rewind while viewing with a media player. The Concert1 program provides the primary live video stream of the entire event. The CDNs are provided with URIs for mapping prior to the event.

The remaining programs—Vip1, Act1, Act2 and Vip2—should be in a running state during the defined start and stop times, with a variance allowed for transitions. A program doesn't start or stop immediately. You need to consider transitions when planning a live event. You can have multiple programs in a running state at the same time, all capturing video. Concert1 and another program will start and stop during the course of the event. To show how you might

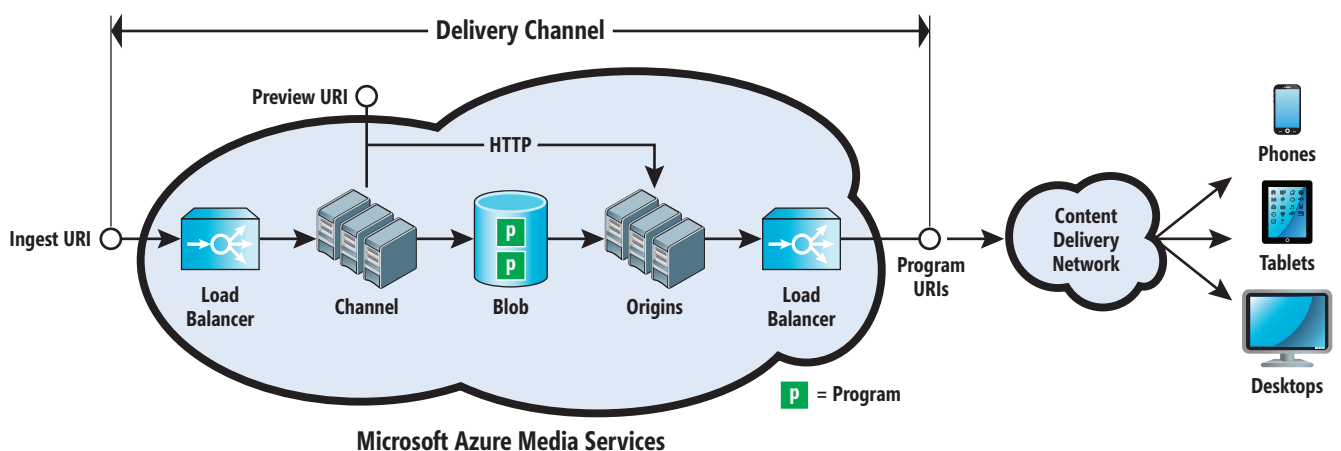


Figure 1 A Delivery Channel Is the Full End-to-End Path of a Live Video Stream

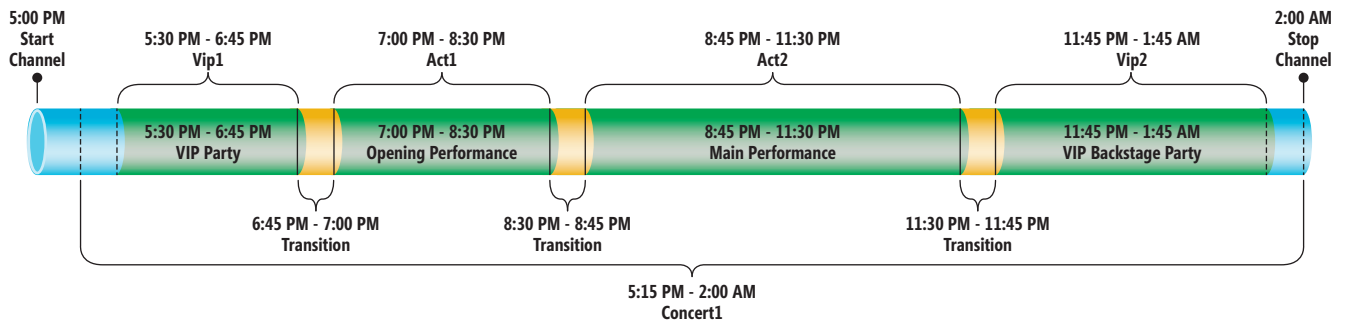


Figure 2 A Live Video Event Mapped to Start at 5 p.m. and End at 2 a.m.

use a transition, you could issue a program start request for Vip2 at 6:45 p.m., then a program stop request for Vip1 at 7 p.m., providing a 15-minute transition time. Therefore, during this 15-minute transition, there could be three programs in a running state.

Having described general concepts and terminology, I'll focus on using the event timeline illustrated in **Figure 2**, mapped onto a user scenario to develop coding examples using the Azure Media Services API.

The Theme: Contoso's Blues Bar

A well-known U.S. venue for up-and-coming musicians, Contoso's Blues Bar has a tight deadline to coordinate an upcoming live concert for a large European band called Contoso and the Shock Wallets. The band's fan base is largely in Europe. The show at Contoso's will be the band's U.S. debut. Therefore, the show managers need to stream the concert live so the band's European fan base can watch on computers and mobile devices.

The CIO of Contoso's Blues Bar calls a meeting and asks his technology team to research and recommend a solution that will meet the following requirements:

- It must be easy to implement.
- It must stream video to a large number of different device configurations.
- It must meet unknown scalability needs.
- The cost is based on what's used during the event, pay-as-you-go.

Sprint One

The Contoso's Blues Bar event planning team spends the next few days defining their user stories. These user stories will be the basis for sprint planning to deliver video streaming of their live concert. During the meeting, the team defines some knowledge gaps that will force a number of spikes—or questions demanding resolution—during sprint one. The team also defines the following list of the larger user stories, typically referred to as Epics, in the standard “as a ... I want ... so that ...” format:

As a fan of Contoso and the Shock Wallets, I want to watch the live concert on my device at the highest video quality possible so that I can enjoy their U.S. debut.

As a fan of Contoso and the Shock Wallets, I want to watch a video of the concert on my device at the highest possible video quality at a later date and time so that I can watch the event at my leisure.

As a Contoso's Blues Bar representative, I want to deliver a live broadcast of our concerts and reduce the expense of delivering

live streaming video from our venue so that we can attract more musicians and customers while saving money.

The user stories and associated spike investigations include:

User Story 1.1: As an event production staffer, I want one or more cameras so that we can capture the live concert.

Spike 1.1.1: *What camera type is being used and what is the output video stream?*

The production staff learns they can purchase high-quality used video cameras to produce a high definition (HD) video/audio stream over a Serial Digital Interface (SDI).

User Story 1.2: As an event production staffer, I want to stream live video to the Internet so that fans can watch a concert.

Spike 1.2.1: *How will the camera video feed get delivered to the production station?*

The production staff learns Contoso Switch Company produces an HD SDI-to-fiber-optics video switch. They can use this switch with up to four cameras.

Spike 1.2.2: *How will that video feed be delivered to the IT department?*

The production staff learns the fiber-optics video switch will output an HD SDI video/audio stream they can connect to an audio/video switcher. They can then control which camera signal is active through a broadcast panel. The live video feed is ultimately sent from the broadcast panel by an HD SDI connection.

User Story 1.3: As an IT staffer, I want to deliver the video stream to Windows, iOS and Android devices so the concert video has the maximum device support.

You need to consider transitions
when planning a live event.

Spike 1.3.1: *What type of video feed am I receiving in order to deliver video to the Internet?*

The HD SDI video feed will be sent to the IT department.

Spike 1.3.2: *What type of video feeds are needed for the target devices?*

The IT staff discovers they can use the following adaptive video protocols to deliver video to each of the target devices:

- Smooth Streaming: Windows 8 and Windows Phone 8
- HLS: iOS and Android
- DASH: Windows 8.1, Windows Phone 8 and Xbox One

Spike 1.3.3: *How am I delivering that video to the Internet in a scalable fashion?*



MS Word Compatible Reporting for the .NET Framework

A Reporting Q&A with Bjoern Meyer
President Text Control, LLC - USA

Text Control is an award-winning Visual Studio Industry Partner and leading vendor of reporting and word processing components for developers of Windows, Web and mobile applications.

Q How do developers typically use Text Control components?

A Our products help thousands of developers add mail merge and reporting functionality to their Windows Forms, WPF and ASP.NET applications. They create elegant and powerful business reports with an easy-to-use, MS Word compatible template editing interface.

Q What sets Text Control Reporting apart from other reporting vendors?

A Text Control Reporting combines the power of a reporting tool and an easy-to-use WYSIWYG word processor that is fully programmable and embeddable in .NET applications. With Text Control Reporting, users can create documents and templates using ordinary Microsoft Word skills.

Q You call Text Control Reporting “Flow Type Layout Reporting”. What is it?

A Banded report designers are limited and not easy to use. Iterating over any data, in any way, across any part of the template is not possible. Using “Flow Type Layout Reporting”, sub-reports for instance are integrated into the document. It is not limited to the report header, the main table and the footer. You can position all kinds of word processing elements in your templates:

- Headers and footers
- Text frames
- Section breaks
- Different page sizes and formatting
- Nested master-detail views
- Images, barcodes and charts

Date	Account	Description	Housing	Transport	Fuel	Meals	Entertainment	Misc	Total
01/05/2014	234452	Business trip	\$220.00	\$1,322.00	\$0.00	\$200.00	\$330.00	\$133.00	\$2,185.00
02/05/2014	234453	Business trip	\$128.00	\$3,322.00	\$0.00	\$200.00	\$320.00	\$22.00	\$3,992.00
03/05/2014	234452	Business trip	\$225.00	\$7,322.00	\$0.00	\$209.00	\$320.00	\$133.00	\$8,209.00
12/05/2014	234452	Business trip	\$0.00	\$3,322.00	\$200.00	\$209.00	\$320.00	\$0.00	\$4,045.00
			\$368.00	\$13,288.00	\$200.00	\$808.00	\$1,270.00	\$286.00	
		Subtotal:							\$18,421.00

Q What is different for developers?

A Text Control Reporting is based on the powerful word processing component TX Text Control. The MS Word compatible template can be merged with a data object (business object) or database content with one line of code. At the same time, Text Control provides a powerful API to customize this merge process completely. The report generation can be fully integrated into the .NET application.

Q You announced a cross-browser HTML5-based Web editor. Tell us more.

A TX Text Control X11 will allow the creation of documents and templates using any operating system with an HTML5 capable browser including Chrome, Firefox, Safari and Internet Explorer. Because the product is being built with pure HTML5 and JavaScript, it will have a zero footprint with no client-side browser plugins required.

For more detailed information and a 30-day trial version, visit www.textcontrol.com/reporting/

The IT staff learns Microsoft has announced a new feature on Azure that specifically addresses streaming live video over the Internet. With some research, the team discovers Azure Media Services can provide the middle layer between their venue and the targeted devices. Most important, they learn the following details about Azure Media Services:

- Once you've signed up for an Azure account and add a Media account, you can create a live streaming channel. A live channel is synonymous to a TV channel. All allocated server resources are dedicated to that channel for the delivery of your programs. A channel may have many programs. A program is the definition of a time slot and the associated asset. An asset is a storage location of the streamed video in Azure Media Services.
- The server allocation ensures redundancy is built into the video path with no single point of failure.
- A live video stream can be received as RTMP or MPEG TS by Azure Media Services via an ingest URI.
- Devices can request a video stream that's natively supported. Microsoft Azure Media Services will ensure the video is packaged in the proper format base on a device-specific URI.
- Support is built into the live streaming origin servers for secure access by a CDN provider, such as Akamai Technologies.

Additional Results of Spike 1.3.1 and 1.3.2: The IT staff chooses an encoder, which can receive an HD SDI video/audio stream. This encoder can use the HD SDI streams to dynamically generate multiple bitrate streams for delivery to Azure Media Services, which provides an ingress URI that accepts RTMP or MPEG TS as an input format.

Sprint Two

After their morning coffee rituals, the developers begin developing code that will facilitate the live streaming event.

User Story 1.4.1: As a developer, I want to create the proper Azure accounts so that I can begin writing code to stream video.

Go to the Azure Web site (azure.microsoft.com), click on the Try for free button and follow the steps. Learn more about setting up an account at bit.ly/1mfaact. During the sign-up process, you can create a Windows Account that's used as the administrator credentials. To create an Azure Media Services account, you must first create an Azure Storage account, as the video assets for a live event will be stored in Blob storage. The Azure Media Services documentation also recommends you create the storage account in the same

datacenter from which you procure the Azure Media Services account. For example, create a storage account and media account in the US-WEST datacenter.

User Story 1.4.2: As a developer, I want to write code to manage a channel so that I can deliver a live streaming event.

It's recommended for production channels that you restrict access by using known values such as your encoders' or authentication tokens' IP address.

Create a base project using Visual Studio 2012. Next, install the NuGet packages for Azure Media Services. Create a function called `CreateContosLiveEvent` and start with a `CloudMediaContext` object, which is always used to manage your live streams:

```
private void CreateContosLiveEvent()
{
    string liveAccount = "yourAzureMediaAccount";
    string liveKey = "yourAzureMediaAccountKey";
    CloudMediaContext LiveServices = new CloudMediaContext(
        liveAccount, // URI created earlier
        liveKey ); // Account name
```

Write the code to ensure an origin service is running. The origin service will deliver the live streams to the CDN providers, as shown in **Figure 3**.

Then you need to write the code to create a channel (shown in **Figure 4**). A channel requires security configured to allow authentication of the inbound video stream source. The source is typically an IP-configured encoder that converts an HDI SDI signal into the required MBR video streams.

Figure 4 Create a Channel for Your Video Feed

```
string channelName = "ContosoBluesChannel";
IChannel channel = LiveServices.FindChannel(channelName);
if (channel != null)
{
    Debug.WriteLine("Channel already exists!");
    return;
}
ChannelSettings settings = new ChannelSettings();
IIPv4 ipv4 = new IIPv4();
// Currently setting IP to 0.0.0.0/0 allows all connections
// Don't do this for production events
ipv4.IP = "0.0.0.0/0";
ipv4.Name = "Allow all connections";
// Protect the ingest URI
settings.Ingest = new IngestEndpointSettings();
settings.Ingest.Security = new SecuritySettings();
settings.Ingest.Security.IPV4AllowList = new List<IIPv4>();
settings.Ingest.Security.IPV4AllowList.Add(ipv4);
// Protect the preview URI
settings.Preview = new PreviewEndpointSettings();
settings.Preview.Security = new SecuritySettings();
settings.Preview.Security.IPV4AllowList = new List<IIPv4>();
settings.Preview.Security.IPV4AllowList.Add(ipv4);
// Create the channel
Task<IChannel> taskChannel = LiveServices.Channels.CreateAsync(
    channelName, "video streaming", ChannelSize.Large, settings);
taskChannel.Wait();
channel = taskChannel.Result;
```

Figure 3 An Origin Service Delivers Live Video Streams to Content Delivery Network Providers

```
string originName = "abcdefg";
IOrigin origin = LiveServices.FindOrigin(originName);
if (origin == null)
{
    Task<IOrigin> originTask = LiveServices.Origins.CreateAsync(originName, 2);
    originTask.Wait();
    origin = originTask.Result;
}
if (origin.State == OriginState.Stopped)
{
    origin.StartAsync().Wait();
}
```



Extreme Performance Linear Scalability

For .NET & Java Apps

(Microsoft Azure Supported)

Cache data, reduce expensive database trips, and scale your apps to extreme transaction processing (XTP) with NCache.

Enterprise Distributed Cache

- Extremely fast & linearly scalable with 100% uptime
- Mirrored, Replicated, Partitioned, and Client Cache
- NHibernate & Entity Framework Second Level Cache

ASP.NET Optimization in Web Farms

- ASP.NET Session State storage
- ASP.NET View State cache
- ASP.NET Output Cache provider

Runtime Data Sharing

- Powerful event notifications for pub/sub data sharing



Download a **FREE** trial!

sales@alachisoft.com

US: +1 (925) 236 3830

www.alachisoft.com

The sample code configures ingest and preview URIs with a Classless Inter-Domain Routing (CIDR) formatted IP address set to "0.0.0.0/0." This allows all IP addresses access to the ingest and preview URIs. It's recommended for production channels that you restrict access by using known values such as your encoders' or authentication tokens' IP address. Use a unique channel name. There will be an exception thrown if there's already a channel with the same name.

Write the code that creates the programs, associated assets and locators, as shown in **Figure 5**. The values used for the names and timelines are related to the values presented in **Figure 2**. The enableArchive flag is set to true during program creation for everything except Concert1. A true value indicates a live video stream is captured during the program's running state and remains persisted with the associated asset for later consumption as VOD. The locator with an access policy of 30 days is created to the asset's manifest file. This provides a URI to the streaming video.

You can obtain an origin URI from a locator, as with an Azure Media Services VOD asset. You would then provide this to the CDN provider of choice. Because the enableArchive flag was set, once a program has been stopped, you can use the same origin URI to deliver the live stream as a VOD asset.

Figure 5 Create the Programs That Will Run During Your Feed

```
// Define the program name, DVR window and estimated duration in minutes.
// NOTE: DVR window value most likely will be removed when service
// reaches public preview.
Tuple<string, int, int>[] programSettings = new Tuple<string, int, int>[]
{
    new Tuple<string,int,int>( "Concert1", 60, 60 ),
    new Tuple<string,int,int>( "Vip1", 75, 75 ),
    new Tuple<string,int,int>( "Act1", 90, 90 ),
    new Tuple<string,int,int>( "Act2", 165, 165 ),
    new Tuple<string,int,int>( "Vip2", 120, 120 ),
};

foreach (Tuple<string, int, int> programSetting in programSettings)
{
    IAsset asset = null;

    // To persist specific program's asset for Video on Demand (VOD) this
    // code tests if the DVR window and Event duration equal in length.
    // If true it sets the enable archive flag to true, which tells the
    // system a program's asset should not be deleted automatically.
    bool enableArchive = programSetting.Item2 == programSetting.Item3;
    try
    {
        // Create the asset that is used to persist the video streamed
        // while the program is running and VOD.
        asset = LiveServices.Assets.Create(programSetting.Item1 + "_asset",
            AssetCreationOptions.None);
        Task<IProgram> taskProgram = channel.Programs.CreateAsync(
            programSetting.Item1,
            "program description",
            // Enable archive note forcing to true for now
            true, // enableArchive,
            // NOTE: DVR window value most likely will be removed
            // This is not used
            TimeSpan.FromMinutes(programSetting.Item2),
            // Estimated duration time
            TimeSpan.FromMinutes(programSetting.Item3),
            asset.Id);
        taskProgram.Wait();
        LiveServices.CreateLocators(asset, TimeSpan.FromDays(30));
    }
    catch (Exception exp)
    {
        Debug.WriteLine(exp.Message);
    }
}
```

The remaining task is to write the code to start a channel and the programs when needed. The request to start a channel signals Azure Media Services to begin allocating resources for the ingress services, redundancy and load balancing. When a channel starts, the time between the starting and running states can take 10 to 20 minutes.

You should start the channel early enough prior to the event so as to not interfere with the ability to view the live video stream. It's also important to note that once your channel has started, billing charges begin to accumulate. If you don't plan on running a 24x7 live channel, you should stop the channel and associated programs when they're not being used. The actual code needed to start a channel is quite simple:

```
// Start channel
Task start = channel.StartAsync();
start.Wait();
```

The code you need to start the program is also straightforward:

```
start = program.StartAsync();
start.Wait();
```

One primary requirement for the event is the ability to control when the channel and each program start and stop. Therefore, the next item up for development is a simple UI that will create the event in its entirety by calling the method `CreateContosLiveEvent`. This permits starting and stopping the channel and each corresponding program as needed.

To close out the scenario, as you've developed the application using Azure Media Services, the other team members have been busy setting up the cameras, running cables, installing a video encoder and configuring CDNs. Finally, the staff performs a series of tests to ensure the end-to-end scenario works.

Once the actual concert is about to happen, a live video stream starts and different user devices, desktops and tablets connect to Azure Media Services to view the live video streams. There are always a few problems uncovered during test runs, but the staff works the kinks out, and all the hard work is done in time to show live video streaming of the concert.

A Daunting Task Made Easier

Hosting live video events at scale can be a daunting task. Before solutions such as Azure Media Services, delivering live streaming video often required significant capital expenditure for infrastructure equipment. Other critical decisions included the quantity of systems needed to support different events that vary in size and scale. Such purchases were often either over- or under-provisioned. Microsoft Azure Media Services responds to that complexity and cost. The Live offering is built atop the Azure Media Services VOD feature set, providing a single, unified platform and APIs for delivering both live and VOD content to the industry at scale. ■

GREGORY PRENTICE is a seasoned software architect with more than 25 years of experience designing and creating applications for various startup companies. He started working for Microsoft developing the following projects: Locadio, Microsoft Hohm and Microsoft Utility Rates Service. Most recently he helped the Microsoft Azure Media Services and Delta Tre teams deliver the 2014 Sochi Olympics live video stream. He's currently an evangelist in the Developer and Platform Evangelism group at Microsoft. Read his blog at blogs.msdn.com/b/greg-prentice.

THANKS to the following Microsoft technical expert for reviewing this article:
Steven Goulet



Develop on Windows, Deploy Everywhere

NOV – .NET UI Platform for Windows, Mac, Silverlight, Linux, iOS and Android.

Q&A with Ivo Milanov, CTO at Nevron Software LLC

Q Why did you develop Nevron Open Vision (NOV)?

A Everybody considers .NET as a framework for portable application development, but .NET did not have a User Interface platform that can work on all Operating Systems. Java and HTML have such a platform for years. We believe that it is time for .NET developers to start coding for all Operating Systems from a single code base—Windows, Mac, Linux, Silverlight, iOS and Android. That is why we developed NOV—it allows you to create User Interfaces that run everywhere, not just Windows.

Q What are the benefits for developers?

A The major one I would consider is Code Once—Run Everywhere. It is a dream come true—no more “How can I do that here like I did it there?”. NOV instantly makes you an UI expert for WinForms, WPF, Silverlight, MonoMac, Xamarin.Mac, Xamarin.iOS and Xamarin.Android development.

Q What are the benefits for project managers and business owners?

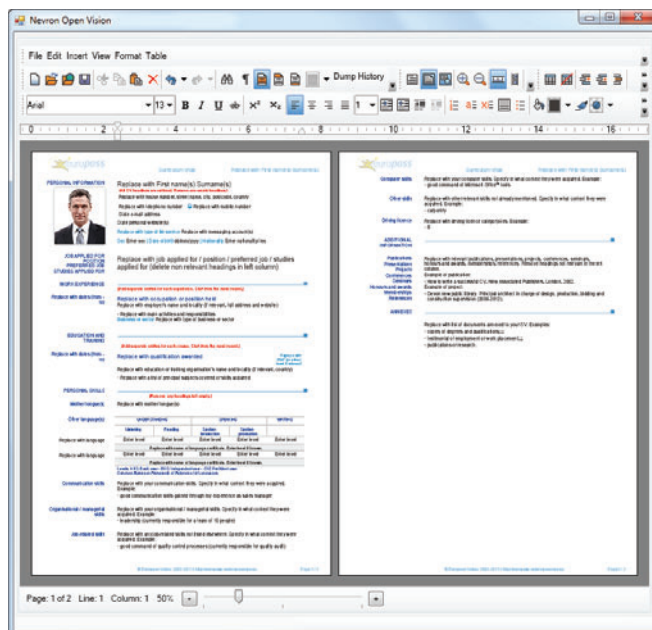
A The major ones according to me are:

- Go to new markets faster
- Develop native apps for multiple markets simultaneously.

We only have 2 Mac developers, but we are now releasing a commercial Text Editor that will rival Microsoft Word on Mac. This is possible because of NOV. The NOV Text editor was developed for more than 5 years in Visual Studio under Windows. We are going to continue its development on Windows and simultaneously release versions for Windows, Mac, Silverlight and Mobile.

Q What controls are you going to add to NOV?

A Currently NOV contains a very extensive UI library, Microsoft Word-like text processor, Gauges and Barcode components. In the near future we’re going to add Grid, Chart, Ribbon and Diagram controls.



Q What sets Nevron apart from other component vendors?

A We may not have a large portfolio of products, but we develop only high-quality and feature-rich solutions. We want to give developers the tools needed to develop commercial quality applications—applications that are competitive on the Enterprise-level. Not many component vendors do that.

Nevron is also the only component vendor that develops a fully managed, HTML 5-like, cross-platform suite of UI components. This is very different from simply making components for specific platforms like the other vendors do. No other vendor can guarantee you 100% identical components, API and features across WinForms, WPF, Silverlight, MonoMac and Xamarin.Mac environments—only Nevron can.

So we are not simply different—we are unique.

For more information, please visit →

www.nevron.com

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

WASHINGTON, D.C.

October 6 – 9, 2014

Washington Marriott at Metro Center



TO BOLDLY
CODE

WHERE NO VISUAL STUDIO LIVE!
HAS CODED BEFORE

THAT'S RIGHT, WE'RE TRANSPORTING
Visual Studio Live! to our nation's capital for the
first time in 21 years. From Oct 6 – 9, 2014,
developers, software architects, engineers and
designers will gather for 4 days of cutting-edge
education on the Microsoft Platform.

Explore HOT TOPICS like:

- Visual Studio 2013 IDE
- MVC 5
- Breeze
- C# 6.0
- Angular
- Entity Framework
- Xamarin
- And Many More!

**YOUR GUIDE TO THE
.NET DEVELOPMENT
UNIVERSE**



**Register by
August 13
and Save \$300!**

Use promo code DCAUG2



Scan the QR code to
register or for more
event details.

CONNECT WITH VISUAL STUDIO LIVE!



twitter.com/vslive – @VSLive



facebook.com – Search “VSLive”



linkedin.com – Join the
“Visual Studio Live” group!

vslive.com/dc

SUPPORTED BY

Microsoft



Visual Studio

msdn
magazine

Visual Studio
MAGAZINE

PRODUCED BY

1105 MEDIA

Washington, D.C. AGENDA AT-A-GLANCE

Visual Studio / .NET Framework	Windows Client	Cloud Computing	Windows Phone	Cross-Platform Mobile Development	ASP.NET	JavaScript / HTML5 Client	SharePoint / Office	SQL Server
--------------------------------	----------------	-----------------	---------------	-----------------------------------	---------	---------------------------	---------------------	------------

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, October 6, 2014 (Separate entry fee required)		
7:30 AM	9:00 AM	Pre-Conference Workshop Registration		
9:00 AM	6:00 PM	MW01 - Workshop: Deep Dive Into Visual Studio 2013, TFS, and Visual Studio Online - <i>Brian Randell</i>	MW02 - Workshop: SQL Server for Developers - <i>Andrew Brust & Leonard Lobel</i>	MW03 - Workshop: Data-Centric Single Page Applications with Angular, Breeze, and Web API - <i>Brian Noyes</i>
6:00 PM	9:00 PM	Dine-A-Round Dinner		

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, October 7, 2014					
7:00 AM	8:00 AM	Registration					
8:00 AM	9:00 AM	Keynote: To Be Announced					
9:15 AM	10:30 AM	T01 - Great User Experiences with CSS 3 - Robert Boedigheimer	T02 - What's New in MVC 5 - Miguel Castro	T03 - New IDE and Editor Features in Visual Studio 2013 - Deborah Kurata		T04 - Creating Universal Windows Apps for Business - Rockford Lhotka	
10:45 AM	12:00 PM	T05 - Using jQuery to Replace the Ajax Control Toolkit - Robert Boedigheimer	T06 - ASP.NET Reloaded: Web Forms vs. MVC vs. Web API - Dino Esposito	T07 - Introduction to In Memory OLTP Using Hekaton in SQL Server 2014 - Kevin Goff		T08 - WPF Data Binding in Depth - Brian Noyes	
12:00 PM	1:30 PM	Lunch - Visit Exhibitors					
1:30 PM	2:45 PM	T09 - Build an Angular and Bootstrap Web Application in Visual Studio from the Ground Up - Deborah Kurata	T10 - What's New in Web API 2 - Miguel Castro	T11 - Making the Most of the Visual Studio Online - Brian Randell		T12 - Moving Web Apps to the Cloud - Eric D. Boyd	
3:00 PM	4:15 PM	T13 - JavaScript for the C# (and Java) Developer - Philip Japikse	T14 - Never Mind the Mobile Web; Here's the Device Web - Dino Esposito	T15 - Cross-platform Dev (iOS, Android and Java) with TFS and Team Explorer Everywhere - Brian Randell		T16 - Solving Security and Compliance Challenges with Hybrid Clouds - Eric D. Boyd	
4:15 PM	5:45 PM	Exhibitor Welcome Reception					

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, October 8, 2014			
7:00 AM	8:00 AM	Registration			
8:00 AM	9:00 AM	Keynote: To Be Announced			
9:00 AM	9:30 AM	Networking Break			
9:30 AM	10:45 AM	W01 - Writing Next Generation JavaScript with TypeScript - <i>Rachel Appel</i>	W02 - A Survey of Two Popular Visual Studio Tools - Web Essentials and NuGet - <i>John Petersen</i>	W03 - Getting Started with Xamarin - <i>Walt Ritscher</i>	W04 - SQL for Application Developers - Attendees Choose - <i>Kevin Goff</i>
11:00 AM	12:15 PM	W05 - Building Single-Page Web Applications Using Kendo UI and the MVVM Pattern - <i>Ben Hoelting</i>	W06 - ASP.NET MVC - Rudiments of Routing - <i>Walt Ritscher</i>	W07 - Getting Started with Windows Phone Development - <i>Nick Landry</i>	W08 - Database Development with SQL Server Data Tools - <i>Leonard Lobel</i>
12:15 PM	1:30 PM	Birds-of-a-Feather Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	W09 - Introduction to AngularJS - <i>John Petersen</i>	W10 - Slice Development Time with ASP.NET MVC, Visual Studio, and Razor - <i>Philip Japikse</i>	W11 - The Great Mobile Debate: Native vs. Hybrid App Development - <i>Nick Landry</i>	W12 - Learning Entity Framework 6 - <i>Leonard Lobel</i>
3:00 PM	4:15 PM	W13 - Creating Angular Applications Using Visual Studio LightSwitch - <i>Michael Washington</i>	W14 - Build Data Driven Web Sites with WebMatrix 3 and ASP.NET Web Pages - <i>Rachel Appel</i>	W15 - Busy .NET Developer's Guide to iOS - <i>Ted Neward</i>	W16 - Team Foundation Server for Scrum Teams - <i>Richard Hundhausen</i>
4:30 PM	5:45 PM	W17 - To Be Announced	W18 - Build Real Time Websites and Apps with SignalR - <i>Rachel Appel</i>	W19 - Busy .NET Developer's Guide to Android - <i>Ted Neward</i>	W20 - Team Foundation Server: Must-Have Tools and Widgets - <i>Richard Hundhausen</i>

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, October 9, 2014					
7:30 AM	8:00 AM	Registration					
8:00 AM	9:15 AM	TH01 - Excel, Power BI and You: An Analytics Superhub - <i>Andrew Brust</i>	TH02 - What's New in WPF 4.5 - <i>Walt Ritscher</i>	TH03 - Visual Studio 2013, Xamarin and Windows Azure Mobile Services: A Match Made in Heaven - <i>Rick Garibay</i>	TH04 - What's New in the .NET 4.5.1 BCL - <i>Jason Bock</i>		
9:30 AM	10:45 AM	TH05 - Big Data 101 with HDInsight - <i>Andrew Brust</i>	TH06 - WPF for the Real World - <i>Brian Lagunas</i>	TH07 - Building Your First Windows Phone 8.1 Application - <i>Brian Peek</i>	TH08 - Essential C# 6.0 - <i>Mark Michaelis</i>		
11:00 AM	12:15 PM	TH09 - Cloud or Not, 10 Reasons Why You Must Know "Web Sites" - <i>Vishwas Lele</i>	TH10 - Deploying WinRT Apps Without the Store - <i>Rockford Lhotka</i>	TH11 - Building Games for Windows and Windows Phone - <i>Brian Peek</i>	TH12 - To Be Announced		
12:15 PM	1:15 PM	Lunch					
1:15 PM	2:30 PM	TH13 - Loosely Coupled Applications with Service Bus and Document-centric Data Stores - <i>Vishwas Lele</i>	TH14 - Creating Cross Platform Games with Unity - <i>Brian Lagunas</i>	TH15 - Visual Studio Cloud Business Apps - <i>Michael Washington</i>	TH16 - Asynchronous Debugging in .NET - <i>Jason Bock</i>		
2:45 PM	4:00 PM	TH17 - Not Your Father's BizTalk Server: Building Modern Hybrid Apps with Windows Azure BizTalk Services - <i>Rick Garibay</i>	TH18 - Use Your .NET Code in WinRT with Brokered Assemblies - <i>Rockford Lhotka</i>	TH19 - Best Practices: Building Apps for Office Using HTML/JavaScript - <i>Mark Michaelis</i>	TH20 - Adventures in Unit Testing: TDD vs. TED - <i>Ben Hoelting</i>		
4:15 PM	5:15 PM	Conference Wrap-Up - <i>Andrew Brust (Moderator), Jason Bock, Ben Hoelting, Rockford Lhotka, & Brian Peek</i>					

Sessions and speakers subject to change

Use Distributed Cache in Microsoft Azure

Iqbal M. Khan and Jeremiah Talkar

Microsoft Azure is rapidly becoming the cloud choice for .NET applications. Besides its rich set of cloud features, Azure provides full integration with the Microsoft .NET Framework. It's also a good choice for Java, PHP, Ruby and Python apps. Many of the applications moving to Azure are high-traffic, so you can expect full support for high scalability. In-memory distributed cache can be an important component of a scalable environment.

This article will cover distributed caching in general and what it can provide.

The features described here relate to general-purpose in-memory distributed cache, and not specifically Azure Cache or NCache for Azure. For .NET applications deployed in Azure, in-memory distributed cache has three primary benefits:

1. Application performance and scalability
2. Caching ASP.NET session state, view state and page output
3. Sharing runtime data with events

This article discusses:

- How distributed caching works
- The different options caching provides
- How to configure the various caching options

Technologies discussed:

Microsoft Azure, Microsoft .NET Framework, Java, PHP, Ruby, Python

Application Performance and Scalability

Azure makes it easy to scale an application infrastructure. For example, you can easily add more Web roles, worker roles or virtual machines (VMs) when you anticipate higher transaction load. Despite that flexibility, data storage can be a bottleneck that could keep you from being able to scale your app.

Unlike a relational database, an in-memory distributed cache scales in a linear fashion.

This is where an in-memory distributed cache can be helpful. It lets you cache as much data as you want. It can reduce expensive database reads by as much as 90 percent. This also reduces transactional pressure on the database. It will be able to perform faster and take on a greater transaction load.

Unlike a relational database, an in-memory distributed cache scales in a linear fashion. It generally won't become a scalability bottleneck, even though 90 percent of the read traffic might go to the cache instead of the database. All data in the cache is distributed to multiple cache servers. You can easily add more cache servers as your transaction load increases. **Figure 1** shows how to direct apps to the cache.

Figure 1 Using In-Memory Distributed Cache in .NET Apps

```
// Check the cache before going to the database
Customer Load(string customerId)
{
    // The key for will be like Customer:PK:1000
    string key = "Customers:CustomerId:" + customerId;
    Customer cust = (Customer)Cache[key];
    if (cust == null)
    {
        // Item not found in the cache; therefore, load from database
        LoadCustomerFromDb(cust);

        // Now, add this object to the cache for future reference
        Cache.Insert(key, cust, null,
            Cache.NoAbsoluteExpiration,
            Cache.NoSlidingExpiration,
            CacheItemPriority.Default, null );
    }
    return cust;
}
```

An in-memory distributed cache can be faster and more scalable than a relational database. **Figure 2** shows some performance data to give you an idea. As you can see, the scalability is linear. Compare this with your relational database or your ASP.NET Session State storage and you'll see the benefit.

Caching ASP.NET Session State, View State and Page Output

Using in-memory distributed cache in Azure also helps with ASP.NET Session State, ASP.NET View State and ASP.NET Output Cache. You'll need to store ASP.NET Session State somewhere. This can become a major scalability bottleneck. In Azure, you can store ASP.NET Session State in a SQL database, Azure Table Storage or an in-memory distributed cache.

A SQL database isn't ideal for storing session state. Relational databases were never really designed for Blob storage, and an ASP.NET Session State object is stored as a Blob. This can cause performance issues and become a scalability bottleneck.

Similarly, Azure Table Storage isn't ideal for Blob storage. It's intended for storing structured entities. Although it's more scalable than a SQL database, it's still not ideal for storing ASP.NET Session State.

An in-memory distributed cache is better suited for storing ASP.NET Session State in Azure. It's faster and more scalable than the other two options. It also replicates sessions so there's no data loss if a cache server goes down. If you store sessions in a separate dedicated caching tier, then Web roles and Web server VMs become stateless, which is good because you can bring them down without losing any session data.

While running ASP.NET Session State in cache is ideal from a performance standpoint, if the cache goes down, your entire app will go down. And, of course, whatever is in your session would

Figure 2 Performance Numbers for a Typical Distributed Cache

Cluster Size	Reads per second	Writes per second
2-node cluster	50,000	32,000
3-node cluster	72,000	48,000
4-node cluster	72,000	64,000
5-node cluster	120,000	80,000
6-node cluster	144,000	96,000

also be gone. The new Redis Cache for Azure session state provider will have a way you can know when these types of issues happen and at least display them to the user in a clean way.

Figure 3 shows how to configure an in-memory distributed cache to store ASP.NET Session State.

Although the ASP.NET MVC framework has removed the need for using ASP.NET View State, the majority of ASP.NET applications haven't yet moved to the ASP.NET MVC framework. Therefore, they still require ASP.NET View State.

ASP.NET View State can be a major bandwidth burden, and cause a noticeable drop in your ASP.NET application response times. That's because ASP.NET View State can be hundreds of kilobytes for each user. It's unnecessarily sent to and from the browser in case of post back. If this ASP.NET View State is cached on the Web server end and a unique identifier is sent to the browser, it can improve response times and also reduce the bandwidth consumption.

An in-memory distributed cache is better suited for storing ASP.NET Session State in Microsoft Azure.

In Azure, where your ASP.NET application is running in multiple Web roles or VMs, the least disruptive place to cache this ASP.NET View State is in an in-memory distributed cache. That way, you can get at it from any Web server. Here's how you can configure the ASP.NET View State for storage in an in-memory distributed cache:

```
<!-- /App_Browsers/Default.browser -->
<browsers>
  <browser refID="Default" >
    <controlAdapters>
      <adapter
        controlType="System.Web.UI.Page"
        adapterType="DistCache.Adapters.PageAdapter"/>
    </controlAdapters>
  </browser>
</browsers>
```

ASP.NET also provides an output cache framework that lets you cache page output that's not likely to change. That way you don't have to execute the page next time. This saves CPU resources and speeds up ASP.NET response time. In a multi-server deployment, the best place to cache page output is within a distributed cache so it will be accessible from all Web servers. Fortunately, ASP.NET Output Cache has a provider-based architecture so you can easily plug in an in-memory distributed cache (see **Figure 4**).

Runtime Data Sharing Through Events

Another reason to consider using in-memory distributed cache in Azure is runtime data sharing. Applications typically do runtime data sharing in the following ways:

1. Polling relational databases to detect data changes
2. Using database events (such as SqlDependency or OracleDependency)
3. Using message queues such as MSMQ

Figure 3 Configure ASP.NET Session State Storage in a Distributed Cache

```
// Check the Cache before going to the database
Customer Load(string customerId)
{
    // The key for will be like Customer:PK:1000
    string key = "Customers:CustomerID:" + customerId;
    Customer cust = (Customer)Cache[key];
    if (cust == null)
    {
        // Item not found in the cach; therefore, load from database
        LoadCustomerFromDb(cust);

        // Now, add this object to the cache for future reference
        Cache.Insert(key, cust, null,
            Cache.NoAbsoluteExpiration,
            Cache.NoSlidingExpiration,
            CacheItemPriority.Default, null );
    }
    return cust;
}
```

These approaches all provide basic functionality, but each has certain performance and scalability issues. Polling is usually a bad idea. This involves many unnecessary database reads. Databases are already a scalability bottleneck, even without additional database events. With the added overhead of database events, databases will choke even more quickly under heavy transaction load.

Message queues specialize in sequenced data sharing and persisting events to permanent storage. They're good for situations where the recipients might not receive events for a long time or where applications are distributed across the WAN. However, when it comes to a high-transaction environment, message queues might not perform or scale like an in-memory distributed cache.

Cache elasticity is an essential aspect of maintaining your in-memory distributed cache.

So if you have a high-transaction environment where multiple applications need to share data at run time without any sequencing and you don't need to persist events for a long time, you might want to consider using an in-memory distributed cache for runtime data sharing. An in-memory distributed cache lets you share data at run time in a variety of ways, all of which are asynchronous:

1. Item level events on update, and remove
2. Cache- and group/region-level events
3. Continuous Query-based events
4. Topic-based events (for publish/subscribe model)

The first three capabilities are essentially different ways to monitor data changes within the cache. Your application registers callbacks for each of these. The distributed cache is responsible for "firing the event" whenever the corresponding data in the cache changes. This results in your application callback being called.

When a specific cached item is updated or removed, there will be an item-level event fired. Cache- and group/region-level events are fired when data in that "container" is added, updated or removed.

Continuous Query consists of search criteria to define a dataset in the distributed cache. The distributed cache fires events whenever you add, update or remove data from this dataset. You can use this to monitor cache changes:

```
string queryString = "SELECT Customers WHERE this.City = ?";
Hashtable values = new Hashtable();
values.Add("City", "New York");

Cache cache = CacheManager.GetCache(cacheName);

ContinuousQuery cQuery = new ContinuousQuery(queryString, values);
cQuery.RegisterAddNotification(new CQItemAddedCallback(cqItemAdded));
cQuery.RegisterUpdateNotification(new CQItemUpdatedCallback(cqItemUpdated));
cQuery.RegisterRemoveNotification(new CQItemRemovedCallback(cqItemRemoved));

cache.RegisterCQ(query);
```

Topic-based events are general purpose, and aren't tied to any data changes in the cache. In this case, a cache client is responsible for "firing the event." The distributed cache becomes something like a message bus and transports that event to all other clients connected to the cache.

With topic-based events, your applications can share data in a publish/subscribe model, where one application publishes data and fires a topic-based event. Other applications wait for that event and start consuming that data once it's received.

Distributed Cache Architecture

High-traffic apps can't afford downtime. For these apps running in Azure, there are three important aspects of in-memory distributed cache:

1. High availability
2. Linear scalability
3. Data replication and reliability

Cache elasticity is an essential aspect of maintaining your in-memory distributed cache. Many in-memory distributed caches achieve elasticity and high availability with the following:

Self-healing peer-to-peer cache cluster: All cache servers form a cache cluster. A self-healing peer-to-peer cluster adjusts itself whenever nodes are added or removed. The more powerful caches form a self-healing peer-to-peer cache cluster, while others form master/slave clusters. Peer-to-peer is dynamic and lets you add or remove cache servers without stopping the cache. Master/slave clusters are limited because one or more of the designated nodes going down hampers cache operations. Some caches such as Memcached don't form any cache cluster and, therefore, aren't considered elastic.

Figure 4 Configure ASP.NET Output Cache for In-Memory Distributed Cache

```
<!-- web.config -->
<system.web>
  <caching>
    <outputCache defaultProvider="DistributedCache">
      <providers>
        <add name="DistributedCache"
            type="Vendor.Web.DistributedCache.DistCacheOutputCacheProvider,
            Vendor.Web.DistributedCache"
            cacheName="default"
            dataCacheClientName="default"/>
      </providers>
    </outputCache>
  </caching>
</system.web>
```

GdPicture.NET 10



- Document viewing, processing, printing and scanning (TWAIN & WIA).
- Reading, writing and converting vector and raster images in more than 90 formats, PDF included.
- OMR, OCR, barcode reading and writing (linear & 2D).
- Annotations for image and PDF within Windows and Web applications.
- Color detection engine for image and PDF compression.

And much more ... 

All-In-One Document Imaging SDK

Royalty-Free Document Imaging Toolkits
for .NET and COM/ActiveX



GdPicture.NET 10 Plugins



Color detection



DICOM image reader



MICR reader

- Full managed PDF support
- Full annotations support for PDF and images
- OCR
- Forms processing
- JBIG2 encoding
- 1D and 2D barcode reading and writing



Try GdPicture.NET 10
FREE for 30 days

www.gdpicture.com

Connection failover: The cache clients are apps running on app servers and Web servers that then access the cache servers. Connection failover capability is within the cache clients. This means if any cache server in the cluster goes down, the cache client will continue working by finding other servers in the cluster.

Dynamic configuration: Both cache servers and cache clients have this capability. Instead of requiring the cache clients to hard-code configuration details, cache servers propagate this information to the cache clients at run time, including any changes.

Caching Topologies

In many cases, you're caching data that doesn't exist in the database, such as ASP.NET Session State. Therefore, losing data can be quite painful. Even where data exists in the database, losing a lot of it when a cache node goes down can severely affect app performance.

Therefore, it's better if your in-memory distributed cache replicates your data. However, replication does have performance and storage costs. A good in-memory cache provides a set of caching topologies to handle different types of data storage and replication needs:

A good in-memory cache provides a set of caching topologies to handle different types of data storage and replication needs.

Mirrored Cache: This topology has one active and one passive cache server. All reads and writes are made against the active node and updates are asynchronously applied to the mirror. This topology is normally used when you can only spare one dedicated cache server and share an app/Web server as the mirror.

Replicated Cache: This topology has two or more active cache servers. The entire cache is replicated to all of them. All updates are synchronous—they're applied to all cache servers as one operation. Read transactions scale linearly as you add more servers. The disadvantage is that adding more nodes doesn't increase storage or update transaction capacity.

Partitioned Cache: This topology has the entire cache partitioned, with each cache server containing one partition. Cache clients usually connect to all cache servers so they can directly access data in the desired partition. Your storage and transaction capacity grows as you add more servers so there's linear scalability. There's no replication, though, so you might lose data if a cache server goes down.

Partitioned-Replicated Cache: This is like a Partitioned Cache, except each partition is replicated to at least one other cache server. You don't lose any data if a cache server goes down. The partition is usually active and the replica is passive. Your application never directly interacts with the replica. This topology provides the benefits of a Partitioned Cache like linear scalability, plus data reliability. There is a slight performance and storage cost associated with the replication.

Client Cache (Near Cache): Cache clients usually run on app/Web servers, so accessing the cache typically involves network traffic. Client Cache (also called Near Cache) is a local cache that keeps frequently used data close to your app and saves network trips. This local cache is also connected and synchronized with the distributed cache. Client Cache can be InProc (meaning inside your application process) or OutProc.

Deploying Distributed Cache in Azure

In Azure, you have multiple distributed cache options, including Microsoft Azure Cache, NCache for Azure and Memcached. Each cache provides a different set of options. These are the most common deployment options for a single region:

1. In-Role Cache (co-located or dedicated)
2. Cache Service
3. Cache VMs (dedicated)
4. Cache VMs across WAN (multi-regions)

In-Role Cache You can deploy an in-role cache on a co-located or dedicated role in Azure. Co-located means your application is also running on that VM and dedicated means it's running only the cache. Although a good distributed cache provides elasticity and high availability, there's overhead associated with adding or removing cache servers from the cache cluster. Your preference should be to have a stable cache cluster. You should add or remove cache servers only when you want to scale or reduce your cache capacity or when you have a cache server down.

The in-role cache is more volatile than other deployment options because Azure can easily start and stop roles. In a co-located role, the cache is also sharing CPU and memory resources with your applications. For one or two instances, it's OK to use this deployment option. It's not suitable for larger deployments, though, because of the negative performance impact.

You can also consider using a dedicated in-role cache. Bear in mind this cache is deployed as part of your cloud service and is only visible within that service. You can't share this cache across multiple apps. Also, the cache runs only as long as your service is running. So, if you need to have the cache running even when you stop your application, don't use this option.

Microsoft Azure Cache and NCache for Azure both offer the in-role deployment option. You can make Memcached run this configuration with some tweaking, but you lose data if a role is recycled because Memcached doesn't replicate data.

Cache Service In a Cache Service, the distributed cache is deployed independent of your service, and offers you a cache-level view. You allocate a certain amount of memory and CPU capacity and create your cache. The benefit of a Cache Service is its simplicity. You don't install and configure any of the distributed cache software yourself. As a result, your cache management effort is reduced because Azure is managing the distributed cache cluster for you. Another benefit is that you can share this cache across multiple applications.

The drawback of a Cache Service is your limited access. You can't control or manipulate the cache VMs within the cluster like you would in an on-premises distributed cache. Also, you can't deploy any server-side code such as Read-through, Write-through,

facebook



Microsoft
SharePoint 2010



Linked in



twitter

SEE THE WORLD AS A DATABASE

ADO.NET ▪ JDBC ▪ ODBC ▪ SQL SSIS ▪ ODATA
MYSQL ▪ EXCEL ▪ POWERSHELL



Microsoft
SQL Server

Linked in

SAP

OData
Open Data Protocol

Salesforce

facebook



Microsoft
SharePoint 2010

amazon
web services

Visual Studio



ODBC

Microsoft
SQL Server

Microsoft
Excel

Microsoft
BizTalk

MySQL

OData

Work With Relational Data, Not Complex APIs or Services

Whether you are a developer using ADO.NET, JDBC, OData, or MySQL, or a systems integrator working with SQL Server or Biztalk, or even an information worker familiar with ODBC or Excel – our products give you bi-directional access to live data through easy-to-use technologies that you are already familiar with. If you can connect to a database, then you will already know how to connect to Salesforce, SAP, SharePoint, Dynamics CRM, Google Apps, QuickBooks, and much more!



Give RSSBus a try today and see what mean:

visit us online at www.rssbus.com to learn more or download a free trial.

rssbus

INTEGRATION YOUR WAY

cache loader and so on. You can't control the number of cache VMs in the cluster because it's handled by Azure. You can only choose among Basic, Standard and Premium deployment options. Microsoft Azure Cache provides a Cache Service deployment option, whereas NCache for Azure does not.

Cache VMs Another option is to deploy your distributed cache in Azure. This gives you total control over your distributed cache. When you deploy your application on Web roles, worker roles or dedicated VMs, you can also deploy the client portion of the distributed cache (the libraries). You can also install the cache client through Windows Installer when you create your role. This gives you more installation options like OutProc Client Cache (Near Cache).

Then you can allocate separate dedicated VMs as your cache servers, install your distributed cache software and build your cache cluster based on your needs. These dedicated cache VMs are stable and keep running even when your application stops. Your cache client talks to the cache cluster through a TCP protocol. For a Cache VM, there are two deployment scenarios you can use:

1. **Deploy within your virtual network:** Your application roles/VMs and the cache VMs are all within the same virtual network. There are no endpoints between your application and the distributed cache. As a result, cache access is fast. The cache is also totally hidden from the outside world and, therefore, more secure.
2. **Deploy in separate virtual networks:** Your application roles/VMs and the cache VMs are in different virtual networks. The distributed cache is in its own virtual network and exposed through endpoints. As a result, you can share the cache across multiple applications and multiple regions. However, you have a lot more control over your Cache VMs than a Cache Service.

In both Cache VM deployment options, you have full access to all cache servers. This lets you deploy server-side code such as read-through, write-through and cache loader, just like you would in an on-premises deployment. You also have more monitoring information, because you have full access to the cache VM. Microsoft Azure Cache doesn't provide the Cache VM option, whereas it's available in NCache for Azure and Memcached.

Microsoft plans to have its managed cache in general availability by July, which will replace the existing shared cache service. It will

most likely not have an Azure portal presence and will require a Windows PowerShell command line to create and manage.

You can install NCache for Azure on dedicated VMs and access it from your Web and Worker Roles. You can also install NCache for Azure on Web and Worker Roles, but that's not a recommended strategy. NCache for Azure doesn't come as a cache service. Microsoft Azure also provides Redis Cache Service, which is available through the management portal.

Cache VMs Across WAN If you have a cloud service that's deployed in multiple regions, consider deploying your Cache VMs across the WAN. The Cache VM deployment in each region is the same as the previous option. However, you have a situation with two additional requirements:

1. **Multi-site ASP.NET sessions:** You can transfer an ASP.NET session from one site to another if a user is rerouted. This is a frequent occurrence for applications deployed in multiple active sites. They may reroute traffic to handle overflows or because they're bringing one site down.
2. **WAN replication of cache:** You can replicate all cache updates from one site to another. This can be an active-passive or active-active multi-site deployment. In active-passive, updates are replicated in one direction. In active-active, they're bidirectional. The cache is kept synchronized across multiple sites through WAN replication, but keep in mind it consumes bandwidth.

Important Caching Features

When you use an in-memory distributed cache, it will handle a lot of data. A basic in-memory distributed cache only provides a hashtable (key, value) interface. An enterprise-level distributed cache may provide the following features as well:

Search features: Instead of always finding data based on a key, it's a lot easier to search the cache based on some other logical data. For example, many distributed caches let you use groups and tags to logically group cached items. Many also let you search the cache through LINQ, which is popular for object searching in the .NET Framework. Some also provide their own Object Query Language (OQL), a SQL-like querying language with which you can search the cache. The ability to search a distributed cache based on attributes other than keys makes the distributed cache look and feel like a relational database. **Figure 5** shows how you might execute such a search.

Read-through and write-through/write-behind: Read-through and write-through handlers simplify your application code because you've moved a large chunk of your persistent code into the cache cluster. Your application simply assumes the cache is its data store. This is another way of reusing code across multiple applications.

The cache calls read-through whenever your application tries to fetch something that isn't in the cache (this is called a "miss"). When a miss happens, the cache calls the read-through handler and it fetches the data from your database (see **Figure 6**). The distributed cache puts it in the cache and returns it to your application.

The cache also calls your write-through handler whenever you update the cache and want it to automatically update the database. Your write-through handler runs on one or more cache servers

Figure 5 Configure a LINQ Search of a Distributed Cache

```
public IList<Customer> GetCustomers(string city)
{
    // Use LINQ to search distributed cache
    IQueryable<Customer> customers = new DistCacheQuery<Customer>(cache);
    try {
        var result = from customer in customers
                     where customer.City = city
                     select customer;

        IList<Customer> prods = new List<Customer>();
        foreach (Customer cust in result) {
            customers.Add(cust);
        }
    }
    return customers;
}
```

Best of the Best: LEADTOOLS Imaging SDKs

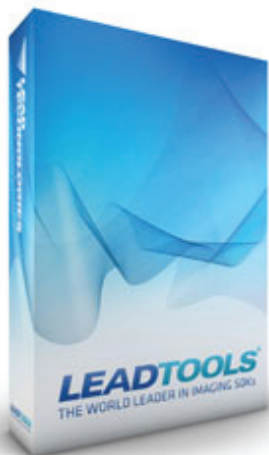
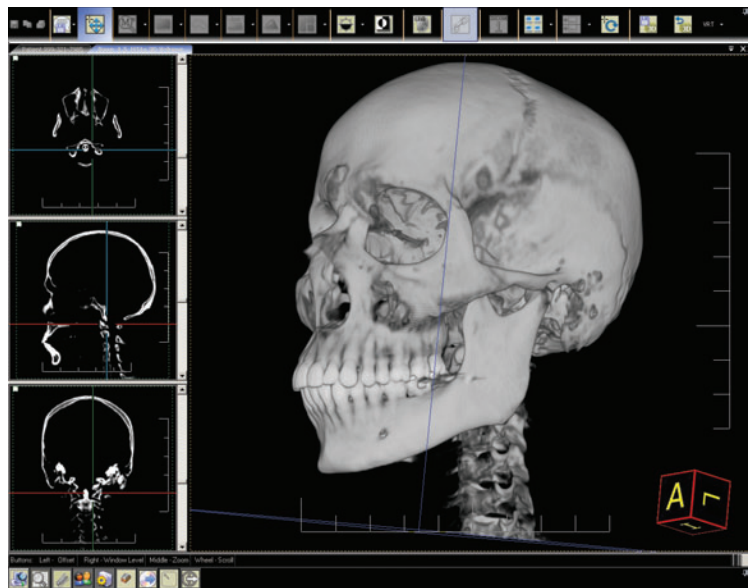
Choosing a software development kit for your application can be a difficult task. You must navigate a harmony of features, ease of use, support, cost and more in an equally hard-to-balance time frame. LEADTOOLS is the World Leader in Imaging SDKs and there are many reasons why.

Unparalleled Experience

Since 1990, LEAD Technologies has been supplying imaging technology to software developers, integrators, contractors and solution providers throughout the United States government, military and most Fortune 1000 companies. LEAD's flagship SDKs, LEADTOOLS, are the most comprehensive and widely used imaging toolkits and are regarded as the gold standard across the globe. Any programmer writing an application for document, medical, multimedia, raster or vector imaging can benefit from the vast development experience packed into the millions of lines of code that comprise LEADTOOLS.

One Stop Shop

If you could summarize LEADTOOLS in one word it would probably be comprehensive. Why? It boasts the most imaging categories on the widest gamut of development platforms including .NET, Win32/64, WinRT, Windows Phone, HTML5/JavaScript, iOS, OS X, Android, Linux and more. With LEADTOOLS, you will be able to create astonishing applications featuring OCR, Forms, Barcode, PDF, DICOM, PACS, Image Processing, MPEG-2 Transport Stream, Multimedia Playback/Capture/Conversion and much more on virtually any platform you can imagine.



Tomorrow's Technology Today

LEAD Technologies is always at the forefront of imaging technology. This means that you can rest assured that when new standards and development platforms are released that LEADTOOLS will have something for you. LEAD also forms strategic partnerships with customers looking for tools targeting cutting edge technology that provide a dual benefit of a custom-tailored solution and expedited time-to-market.

Personal Touch

We software developers sometimes get a bad rap as being anti-social cave dwellers, but we're people too! LEAD Technologies gets that and has utilized the customer feedback and suggestions over the decades to make LEADTOOLS incredibly programmer-friendly. On top of that, free technical support over email, chat and phone is available and LEAD's developer support team loves to help you get the most out of LEADTOOLS with custom-made example projects and pointers to help you along the way.

To learn more please visit our website →

www.leadtools.com

Figure 6 Read-Through Handler for a Distributed Cache

```
public class SqlReadThruProvider : IReadThruProvider
{
    // Called upon startup to initialize connection
    public void Start(IDictionary parameters) { ... }

    // Called at the end to close connection
    public void Stop() { ... }

    // Responsible for loading object from external data source
    public object Load(string key, ref CacheDependency dep)
    {
        string sql = "SELECT * FROM Customers WHERE ";
        sql += "CustomerID = @ID";
        SqlCommand cmd = new SqlCommand(sql, _connection);
        cmd.Parameters.Add("@ID", System.Data.SqlDbType.VarChar);

        // Extract actual customerID from "key"
        int keyFormatLen = "Customers:CustomerID:".Length;
        string custId = key.Substring(keyFormatLen, key.Length - keyFormatLen);
        cmd.Parameters["@ID"].Value = custId;

        // Fetch the row in the table
        SqlDataReader reader = cmd.ExecuteReader();

        // Copy data from "reader" to "cust" object
        Customers cust = new Customers();
        FillCustomers(reader, cust);

        // Specify a SqlCacheDependency for this object
        dep = new SqlCacheDependency(cmd);
        return cust;
    }
}
```

in the cluster and talks to your database. If the write-through is called asynchronously after a delay and not as part of the cache update transaction, however, this operation is called write-behind. Write-behind improves application performance because you don't have to wait for the database update to be completed.

Synchronize cache with relational database: Most data within a distributed cache comes from your application database. That means there are now two copies of the data, the master copy in the database and one in the distributed cache. If you have

Figure 7 Use CacheDependency to Manage Relationships in the Cache

```
public void CacheCustomerAndOrder(Customer cust, Order order)
{
    Cache cache = HttpRuntime.Cache;

    // Customer has one-to-many with Order. Cache customer first
    // then cache Order and create CacheDependency on customer
    string custKey = "Customer:CustomerId:" + cust.CustomerId;
    cache.Add(custKey, cust, null,
        Cache.NoAbsoluteExpiration,
        Cache.NoSlidingExpiration,
        CacheItemPriority.Default, null);

    // CacheDependency ensures order is removed if Cust updated or removed
    string[] keys = new string[1];
    keys[0] = custKey;
    CacheDependency dep = new CacheDependency(null, keys);
    string orderKey = "Order:CustomerId:" + order.CustomerId
        + ":ProductId:" + order.ProductId;

    // This will cache Order object with CacheDependency on customer
    cache.Add(orderKey, order, dep,
        absoluteExpiration,
        slidingExpiration,
        priority, null);
}
```

applications directly updating data in the database, but don't have access to your in-memory distributed cache, you end up with stale data in the cache.

Some distributed caches provide database synchronization to ensure the cache never has stale data. This synchronization is either event-driven (using database notifications such as `SqlDependency`) or polling-based. Event-driven is closer to real time, but has more overhead because it creates a separate `SqlDependency` in the database server for each cached item. Polling is more efficient because one database call can synchronize thousands of items. But there's usually a delay in synchronization, in order to avoid flooding the database with unnecessary polling calls. So your choice is between closer to real-time database synchronization or more efficient polling-based synchronization with a slight delay.

Handling relational data in a distributed cache: An in-memory distributed cache is a key-value object store, but most data cached within comes from a relational database. This data has one-to-many, one-to-one and many-to-many relationships. Relational databases provide referential integrity constraints and cascaded updates and deletes to enforce these relationships. The cache needs something similar as well.

`CacheDependency` lets you have one cached item depend on another. If the other cached item is updated or deleted, the original cached item is automatically deleted. This operates like a cascading delete. It's useful for handling one-to-one and one-to-many relationships between objects in the cache. Some in-memory distributed caches have implemented this feature as well. **Figure 7** shows how you would configure `CacheDependency`.

Wrapping Up

Microsoft Azure is a powerful cloud platform and a scalable environment. An in-memory distributed cache can be an important component of this environment. If you've written your application in the .NET Framework, then you should consider using a .NET distributed cache. If it's in Java, you have various Java-based distributed caching solutions. If you have a combination of .NET and Java applications, use a distributed cache that supports both and provides data portability. Most caches are totally focused on .NET or Java, although some do support both.

For PHP, Ruby, Python and other applications, you can use Memcached. This supports all these environments. Memcached isn't an elastic cache and, therefore, has high-availability and data-reliability limitations. Either way, keep in mind that a distributed cache is a sensitive part of your production environment. Therefore, you must evaluate all available caching solutions in your environment thoroughly and select one that best meets your needs. ■

IQBAL KHAN is the technology evangelist for Alachisoft, which provides Ncache distributed cache for .NET and Microsoft Azure. You can reach him at iqbal@alachisoft.com.

JEREMIAH TALKAR is a Microsoft Azure tech evangelist within the Microsoft Developer Platform Evangelism Corporate Team with 26 years of experience in product engineering, consulting and sales. Reach him at jtalkar@microsoft.com.

THANKS to the following Microsoft technical experts for reviewing this article: Scott Hunter, Masashi Narumoto and Trent Swanson

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

REDMOND

August 11-15, 2014

Microsoft HQ

Redmond, WA

THE IT CLASSROOM ON CAMPUS

TechMentor is returning to Microsoft Headquarters for 5 days of information-packed sessions and workshops! Surrounded by your fellow IT professionals, you will receive immediately usable education that will keep you relevant in the workforce.

TECHMENTOREVENTS.COM/REDMOND



Tracks Include:

- ✦ Windows PowerShell
- ✦ System Center
- ✦ Windows Server
- ✦ Security and Ethical Hacking
- ✦ Advanced Group Policy
- ✦ Hybrid Cloud and Virtualization
- ✦ Jack of All Trades

Register Today!

Use Promo Code TMRAUG1



Scan QR Code
to register or for
more details.

EVENT SPONSOR

Microsoft

GOLD SPONSORS



SUPPORTED BY



PRODUCED BY





Solving Sudoku Puzzles Using the MSF Library

A Sudoku puzzle is an example of what's called a constraint satisfaction problem (CSP). One way to tackle CSPs programmatically is to use the Microsoft Solver Foundation (MSF) library. Although it's highly unlikely you'll ever need to solve a Sudoku puzzle as part of your normal work responsibilities, there are at least three reasons why you might want to read this article. First, the techniques presented here can be used to solve many realistic problems. Second, this article will introduce you to the MSF library and its capabilities. And third, you might just find that solving Sudoku puzzles programmatically is interesting and entertaining.

To get an idea of where this article is headed, take a look at the demo program in **Figure 1**. The demo console application begins by setting up and displaying the data for a typical Sudoku puzzle. Next, it programmatically defines constraints (required conditions) that are common to all Sudoku puzzles, and then sets up the data constraints that are specific to the puzzle. Then, behind the scenes, the demo uses the MSF library to solve the puzzle. The demo concludes by displaying the solution.

This article assumes you have at least intermediate-level programming skills and a vague idea of what Sudoku puzzles are, but does not assume you know anything about constraint satisfaction problems or the MSF library. The demo program is coded using C# but you should be able to refactor the demo to other .NET languages without too much trouble. All the code is presented here and it is also available in the code download that accompanies this article at msdn.microsoft.com/magazine/msdnmag0814. All normal error checking has been removed to keep the main ideas clear.

The Microsoft Solver Foundation Library

The MSF version 3.1 library is available as a separate code download. The location of the download has tended to move around over time but I found it at bit.ly/1vayGh1. I prefer to use 32-bit libraries when experimenting so I clicked on that link, which gave me the option to run or save the MicrosoftSolverFoundation.msi installation package. I selected the Run option.

The installation wizard tells you you're installing the Express Edition. MSF originally came in two versions, a free Express Edition and a for-purchase enterprise edition, but the enterprise edition has been discontinued. The MSF library is essentially no longer being actively developed, but the current 3.1 version works fine for me. After the quick but somewhat clunky installation process finishes, the

key library file `Microsoft.Solver.Foundation.dll` is copied to your machine in directory `C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\NETFramework\v4.0`.

To create the demo program, I launched Visual Studio and selected the C# console application program template and named it `SudokuSolver`. The demo has no significant Microsoft .NET Framework version dependencies so any relatively recent version of Visual Studio should work. After the template code loaded, in the Solution Explorer window I renamed file `Program.cs` to `SudokuProgram.cs` and Visual Studio then automatically renamed class `Program`. The overall structure of the demo program, with a few minor edits to save space, is presented in **Figure 2**.

At the top of the Visual Studio template-generated source code, I removed all unnecessary using statements except for the one that references the top-level System namespace. Next, I added a reference to the MSF library DLL file and then added a using statement that references the library to bring it into scope.

Almost all of the work is performed inside method `Main`. Two helper methods, `AddDataConstraints` and `NumberSolutions`, are

```
file:///F:/Data/MSDNMagazine/SudokuSolver/bin/Debug/Sud...
Begin MS Solver Foundation Sudoku demo
The problem is:
-- 6 2 -- -- 8 --
-- 8 9 7 -- -- --
-- 4 8 1 -- 5 --
-- 7 -- 6 -- -- 2
6 -- -- 5 -- --
-- 2 -- 4 7 1 --
-- 3 -- 2 8 4 --
-- 5 -- -- 1 2 --
Creating generic row constraints
Creating generic column constraints
Creating generic sub-cube constraints
Creating problem specific data constraints
Solving. . .
There is/are 1 Solution(s)
7 1 6 2 3 5 9 8 4
5 2 8 9 7 4 3 1 6
3 9 4 8 1 6 5 2 7
8 4 5 1 6 3 7 9 2
2 7 1 4 8 9 6 3 5
6 3 9 7 5 2 8 4 1
9 8 2 6 4 7 1 5 3
1 6 3 5 2 8 4 7 9
4 5 7 3 9 1 2 6 8
End Solver Foundation Sudoku demo
```

Figure 1 Sudoku Using the Microsoft Solver Foundation

Code download available at msdn.microsoft.com/magazine/msdnmag0814.



YOUR .NET Resources



Visual Studio[®]
MAGAZINE

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ONLINE | NEWSLETTERS | PRINT | CONFERENCES

Figure 2 Overall Program Structure

```
using System;
using Microsoft.SolverFoundation.Services;
namespace SudokuSolver
{
    class SudokuProgram
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Begin MS Solver Foundation Sudoku demo");
            Console.WriteLine("The problem is: ");
            int[][] data = new int[9][];

            data[0] = new int[] { 0, 0, 6, 2, 0, 0, 0, 8, 0 };
            data[1] = new int[] { 0, 0, 8, 9, 7, 0, 0, 0, 0 };
            data[2] = new int[] { 0, 0, 4, 8, 1, 0, 5, 0, 0 };

            data[3] = new int[] { 0, 0, 0, 0, 6, 0, 0, 0, 2 };
            data[4] = new int[] { 0, 7, 0, 0, 0, 0, 0, 3, 0 };
            data[5] = new int[] { 6, 0, 0, 0, 5, 0, 0, 0, 0 };

            data[6] = new int[] { 0, 0, 2, 0, 4, 7, 1, 0, 0 };
            data[7] = new int[] { 0, 0, 3, 0, 2, 8, 4, 0, 0 };
            data[8] = new int[] { 0, 5, 0, 0, 0, 1, 2, 0, 0 };

            // all program logic here

            Console.WriteLine("End Solver Foundation Sudoku demo");
            Console.ReadLine();
        }

        static void AddDataConstraints(int[][] data,
            Model model, Decision[][] grid) { . . . }

        static int NumberSolutions(SolverContext problem) { . . . }
    } // ns
}
```

just conveniences to keep the code inside Main a bit tidier. After a preliminary begin-message, the demo sets up the Sudoku puzzle data in an array-of-arrays style matrix.

Unlike many languages, C# supports a true two-dimensional array, but as you'll see, the array-of-arrays approach is easier to work with. Even if you're an experienced programmer, if you don't often work with numerical programming, you may not be familiar with matrix coding techniques. The demo data matrix is represented in **Figure 3**. Data can be accessed by individual cell, or by an entire row, but not by an entire column. For example, `data[0][2]` is the cell at row 0 and column 2 and has value 6, and `data[1]` references the entire second row. There's no convenient way to access a column. The blank cells in **Figure 3** actually have 0 values because C# automatically initializes integer-array cells to all 0s.

Setting up the Problem

After the problem data matrix has been created, the demo program displays the values to the shell:

```
for (int r = 0; r < 9; ++r) {
    for (int c = 0; c < 9; ++c) {
        if (data[r][c] == 0)
            Console.WriteLine(" ");
        else
            Console.Write(" " + data[r][c]);
    }
    Console.WriteLine("");
}
```

Here, the loop limits are hardcoded as 9. In my opinion, especially with demo programs, sometimes it's better to keep

[0]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			6	2				8	
[1]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			8	9	7				
[2]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			4	8	1		5		
[3]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>					6				2
[4]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		7							3
[5]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	6			5					
[6]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			2		4	7	1		
[7]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			3		2	8	4		
[8]	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		5				1	2		
		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

Figure 3 The Problem Data Matrix

things simple rather than make code more general. Next, the demo makes the first use of MSF code:

```
SolverContext problem = SolverContext.GetContext();
Model model = problem.CreateModel();
```

```
Decision[][] grid = new Decision[9][];
for (int r = 0; r < 9; ++r)
    grid[r] = new Decision[9];
```

Working with the MSF library has a somewhat unusual feel because the code was developed in a hybrid research-development environment. You can think of the first two lines as a magic incantation to create a CSP object. Rather than working with a single object, as you're likely used to, the MSF library tends to use multiple objects such as the problem and model objects here.

Unlike many languages, C# supports a true two-dimensional array, but as you'll see, the array-of-arrays approach is easier to work with.

The grid object is an array-of-arrays style matrix where each cell is a Decision object. You can think of a Decision object as an encapsulation of an answer. Or put another way, to solve a Sudoku puzzle you need to determine $9 \times 9 = 81$ values. Each of these values is represented by a Decision object and the demo stores them in a matrix.

At this point, the grid matrix has uninstantiated Decision objects. The demo instantiates each cell like so:

```
for (int r = 0; r < 9; ++r)
    for (int c = 0; c < 9; ++c)
        grid[r][c] = new Decision(Domain.IntegerRange(1, 9),
            "grid" + r + c);
```

```
for (int r = 0; r < 9; ++r)
    model.AddDecisions(grid[r]);
```

The Decision constructor accepts two parameters. The first describes the data type of an answer. Here, each answer is an integer between 1 and 9. The second parameter is a non-optional name as a string. These names must be unique, so the demo programmatically assigns names "grid00," "grid01" and so on to each of the 81 Decision objects. After the Decision objects have been instantiated,

they must be added to the Model object. The AddDecisions method can accept a single Decision object or an array of Decision objects, so the demo passes the nine rows of the grid matrix. An alternative would be to use two nested loops, like this:

```
for (int r = 0; r < 9; ++r)
    for (int c = 0; c < 9; ++c)
        model.AddDecisions(grid[r][c]);
```

Adding Generic Constraints

There are three sets of generic constraints that are common to all Sudoku puzzles.

Visual Studio **LIVE!**

EXPERT SOLUTIONS FOR .NET DEVELOPERS

YOUR GUIDE TO THE .NET DEVELOPMENT UNIVERSE

REDMOND 2014

**August 18 – 22 | Microsoft Headquarters
Redmond, WA**



Tracks Include:

- Visual Studio/.NET Framework
- Windows Client/Phone
- JavaScript/HTML5 Client
- ASP.NET
- Cloud Computing
- Cross-Platform Mobile Development
- SharePoint
- SQL Server

SET YOUR COURSE FOR 127.0.0.1!

From August 18 – 22, 2014, developers, software architects, engineers and designers will land in Redmond, WA at the idyllic Microsoft headquarters for 5 days of cutting-edge education on the Microsoft Platform.

Sessions are filling up quickly – Register Today!

Use promo code REDAUG1



Scan the QR code to
register or for more
event details.

vslive.com/redmond

EVENT SPONSOR
Microsoft

PLATINUM SPONSOR
 **esri**

SPONSOR
 **LogiGear**
Software Testing

SUPPORTED BY
 **Visual Studio**

 **msdn**
magazine

Visual Studio
MAGAZINE

PRODUCED BY
 **1105 MEDIA**

First, the values in each row must all be different. Second, the values in each column must all be different. And third, the values in each 3x3 sub-cube must all be different. Taking care of the first set of constraints is easy:

```
Console.WriteLine("Creating generic row constraints");
for (int r = 0; r < 9; ++r)
    model.AddConstraint("rowConstraint" + r, Model.AllDifferent(grid[r]));
```

The `AddConstraint` method accepts a constraint name followed by a constraint. Here, the names are "rowConstraint0", "rowConstraint1" and so on. The demo uses the `AllDifferent` method to create a constraint. In words, for each of the nine rows, add a constraint that all the values in the row must be different.

Adding the generic column constraint requires a little bit more effort:

```
Console.WriteLine("Creating generic column constraints");
for (int c = 0; c < 9; ++c)
{
    for (int first = 0; first < 8; ++first)
        for (int second = first + 1; second < 9; ++second)
            model.AddConstraint("colConstraint" + c + first + second,
                                grid[first][c] != grid[second][c]);
}
```

Because an entire column cannot be accessed directly, the demo works on each column separately. For column 0, the first time through the two inner nested loops sets the constraint named "colConstraint001" as `grid[0][0] != grid[1][0]`. The second iteration sets "colConstraint002" as `grid[0][0] != grid[2][0]`. To summarize, the `AllDifferent` method accepts a set of Decision objects as an array and behind the scenes generates explicit inequalities. In situations where your Decision objects aren't in an array (as in the column values), you must explicitly generate the inequalities.

By far the trickiest part of the demo program is setting up the constraints that specify all the values in each of the nine 3x3 sub-cubes must be different. That code is presented in **Figure 4**. Bear with me—the code isn't nearly as complex as it appears.

Consider the sub-cube in the lower-left corner of **Figure 3**. The necessary constraints for this sub-cube are:

```
grid[6][0] != grid[6][1]
grid[6][0] != grid[6][2]
grid[6][0] != grid[7][0]
...
grid[8][1] != grid[8][2]
```

There are $8 + 7 + 6 + \dots + 1 = 36$ constraints for this sub-cube, and so there are $9 * 36 = 324$ total inequalities for the nine sub-cube constraints. Now, it would be possible to list each one using copy-paste and some patience, but a programmatic approach is quicker.

Figure 4 Setting up the Sub-Cube Constraints

```
Console.WriteLine("Creating generic sub-cube constraints");
for (int r = 0; r < 9; r += 3) {
    for (int c = 0; c < 9; c += 3) {
        for (int a = r; a < r + 3; ++a) {
            for (int b = c; b < c + 3; ++b) {
                for (int x = r; x < r + 3; ++x) {
                    for (int y = c; y < c + 3; ++y) {
                        if ((x == a && y > b) || (x > a))
                        {
                            model.AddConstraint("cubeConstraint" + a + b + x + y,
                                                    grid[a][b] != grid[x][y]);
                        }
                    } // y
                } // x
            } // b
        } // a
    } // c
} // r
```

In the code, the two outer loops establish an upper-left corner of each sub-cube. In the four innermost loops, cells are represented as `grid[a][b] != grid[x][y]`. If you look just at the indices in the example and image, which are just ordinary integers, you get:

```
60 and 61
60 and 62
...
81 and 82
```

Notice that in each case, there is an inequality constraint when $ab < xy$. The innermost four loops iterate over *a*, *b*, *x*, and *y* to generate all possible pairs of indices and the if-then condition creates a constraint on `grid[a][b]` and `grid[x][y]` only when $ab < xy$.

Adding Problem-Specific Constraints

After the generic constraints have been created and added to the model, the demo program adds the constraints that define the specific Sudoku puzzle. The demo puzzle has 27 fixed values, so you could use brute force, like so:

```
model.AddConstraint("v02", grid[0][2] == 6);
model.AddConstraint("v03", grid[0][3] == 2);
...
model.AddConstraint("v86", grid[8][6] == 2);
```

There's nothing wrong with a brute force approach, but because the data already has been placed into a matrix, it's easy to transfer the values using a program-defined helper method call like this:

`AddDataConstraints(data, model, grid);`

where the helper is defined as:

```
static void AddDataConstraints(int[][] data, Model model, Decision[][] grid)
{
    for (int r = 0; r < 9; ++r) {
        for (int c = 0; c < 9; ++c) {
            if (data[r][c] != 0) {
                model.AddConstraint("v" + r + c,
                                    grid[r][c] == data[r][c]);
            }
        }
    }
}
```

Notice the strange coupling between the Model and Decision objects. Library code written by developers for developers would likely have referenced the Decision (answer) objects along the lines of `model.grid[r][c]`. The unusual style of the MSF library took me about three examples to get comfortable with it.

Solving the Puzzle

With everything in place, the demo program can solve the puzzle with this code:

```
Console.WriteLine("Solving. . .");
int numSolutions = NumberSolutions(problem);
Console.WriteLine("There is/are " + numSolutions + " Solution(s)");
Solution solution = problem.Solve();
```

The `NumberSolutions` method is a program-defined helper that's called to check if there are zero solutions (which means conflicting constraints were somehow defined) or more than one solution:

```
static int NumberSolutions(SolverContext problem)
{
    int ct = 0;
    Solution soln = problem.Solve();
    while (soln.Quality == SolverQuality.Feasible) {
        ++ct;
        soln.GetNext();
    }
    return ct;
}
```

The MSF `Solve` method does just that, placing the actual answers into the Decision objects, which are in the Model object, which is

associated by reference to the SolverContext object. As a side effect, the Solution object has a Quality field that can be one of eight values, including Feasible and Infeasible. The Solution GetNext method doesn't return an explicit value but does modify the Quality field. Don't blame me for the MSF design.

The information presented here should allow you to use the MSF library to solve many practical combinatorial optimization problems you might come across.

The Sudoku demo concludes by displaying the answers stored in the Decision objects inside the grid matrix:

```
for (int r = 0; r < 9; ++r) {
    for (int c = 0; c < 9; ++c) {
        double v = grid[r][c].GetDouble();
        Console.WriteLine(" " + v);
    }
    Console.WriteLine("");
}

Console.WriteLine("End Solver Foundation Sudoku demo");
Console.ReadLine();
} // Main
```

The GetDouble method extracts the answer value from the Decision object. Recall that answers were defined to be integers in the range 1 to 9. However, there's no GetInteger method so answer values are implicitly cast to type double. Because they all end with point-zero, when displayed they appear as integers even though they are type double.

Wrapping Up

The particular type of CSP described in this article is really a combinatorial optimization problem. That is, the goal is to find the combination of values that has the fewest constraint errors. The information presented here should allow you to use the MSF library to solve many practical combinatorial optimization problems you might come across.

I have a confession. My sister's boyfriend introduced me to Sudoku on a family trip to Palm Springs a few years ago. He consistently beats me whenever we compete for time on Sudoku or crossword puzzles or anything else. He doesn't know it's possible to solve any Sudoku puzzle with this code. I'm looking forward to our next family trip. I'll have my laptop. ■

Dr. James McCaffrey works for Microsoft Research in Redmond, Wash. He has worked on several Microsoft products including Internet Explorer and Bing. Reach him at jammc@microsoft.com.

THANKS to the following technical expert for reviewing this article:
Kirk Olynyk (Microsoft Research)

msdnmagazine.com

MSDN Magazine Online



It's like **MSDN Magazine**—only better. In addition to all the great articles from the print edition, you get:

- Code Downloads
- The **MSDN Magazine Blog**
- Digital Magazine Downloads
- Searchable Content

All of this and more at
msdn.microsoft.com/magazine

msdn
magazine



Unwritten Rules

The software industry, like modern society in general, is guided by many written rules, and it's accumulating more all the time. But the older and ~~wiser~~ ~~less stupid~~ more cynical I get, the more I see that both society and the software industry are actually governed more by unwritten rules rather than written ones.

The written rule says: "Thou shalt not do X!" The unwritten rule says, "Yes, you can do X, at least under certain circumstances. You sort of have to in order to get the job done." The former addresses the world as the writer wishes it were; the latter as it actually is.

Complicated? Sure. Nuanced? Definitely. Trouble? Often. But observe the next time an airline pilots or nurses union stages a work-to-rule job action instead of a strike. Without the lubrication of these unwritten rules, their workplaces quickly grind to a gridlocked halt. And so would society if we didn't follow unwritten rules instead of written ones.

Charting a course through these minefields, knowing when to follow the written rules and when to ignore them in favor of the unwritten ones, is what software engineering is about.

Consider Michael Pineda, the New York Yankees pitcher who in April received a 10-game suspension for using pine tar on his fingers to better grip the ball. The written rule says that pitchers can't use pine tar or any other substance. The unwritten rule says that a little pine tar is OK in cold weather, as long as you're discreet about it. Pineda was suspended not so much for using pine tar, but for doing it so blatantly that the umpires had to take notice (you can read about it at foxs.pt/1i3R3fz).

To take an example from our industry, object-oriented programmers have long declared global variables to be taboo. You should be shot for even thinking about them, right? On the other hand, what's a class static, except for a global dressed up in wrappers to make it politically correct? Programmers often hang them from an object class named "Globals." When there's only one instance

of something in a program, and it's an integer, how many layers of abstraction and managers and services do you want me to wrap it up in to allow you to feel virtuous? As your blood pressure sky-rockets from reading that last sentence, I'll rephrase the question: How many layers are you willing to pay me for, when you could use that money for something else? Your check is your answer. And it's probably different from your first reaction two sentences ago.

Sometimes the written rule consists solely of writing down the pre-existing unwritten rule. Hardwiring a string, such as a file name, directly into your code is usually a bad idea; but sometimes you have to do it. Give that construction a virtuous-sounding label and now you're being good.

"We're following the 'Well Known Name' design pattern," says the geek. "Good, good," says the non-technical boss. "I'm glad you're using design patterns instead of that cowboy coding thing I've heard about."

Guys, you're hardwiring in a string. Do that when you need to, with your eyes wide open, when you've thought it out carefully and know that the benefits outweigh the costs. Soothe your uneasy feelings with the lie that you'll come back and fix it someday when you get the time. But don't think you're changing what it does by dressing up its name. A bug hurts and angers your user no less when you call it an issue (see msdn.microsoft.com/magazine/ff955613).

Charting a course through these minefields, knowing when to follow the written rules and when to ignore them in favor of the unwritten ones, is what software engineering is about. There are few, absolute, "thou shalt not" rules in this business (although not scrolling a marquee string in your browser's status bar comes darn close). The one thing I always tell my clients and students is, "Thou shalt think. If that's hard, then thou shalt pay me to help you think."

I'm curious to hear your favorite unwritten rules of software development—or anything else, for that matter. Send them to me at dave@rollthunder.com.

I'm sure that some pedantic geek will now say that because I've written some rules here, they aren't unwritten any more. To which I reply: You're breaking *my* unwritten rule against nit-picking your hyperbolic back-page columnist. That's worth a 10-game suspension, don't you think? ■

DAVID S. PLATT teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.



.NET Tools for the Professional Developer

Data Visualization, User Interface, Reporting and Spreadsheet Controls Experts

ar

ActiveReports 8

Create sophisticated, fast and powerful reports



Studio Enterprise

Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers

sp

Spread Studio

Flexible and familiar spreadsheet architecture, advanced charting and powerful formula library



Download your free trials at
ComponentOne.com

EXPECT MORE. GET MORE.

ComponentOne
a division of GrapeCity®

© 2014 GrapeCity, Inc. All rights reserved. All other product and brand names are trademarks and/or registered trademarks of their respective holders.



BIG DATA MADE EASY FOR .NET DEVELOPERS

SYNCFUSION PROVIDES ESSENTIAL TOOLS FOR BIG DATA AND PREDICTIVE ANALYTICS.

- ★ Easy installation with very little configuration.
- ★ All necessary tools are packaged together.
- ★ Build on top of open source tools that simplify the development on Windows and cloud deployment.
- ★ Our tools provide the missing pieces to integrate big data solutions with .NET.

syncfusion.com/bigdata



Contact us to learn more:
1-888-9-DOTNET
sales@syncfusion.com

